****

**Render -> to deploy server backend**

**Vercel -> to deploy fronted**

**$vercel login**

**$vercel**

**Web Version VS Desktop Version IDE**

When comes it comes to desktop version – smooth access, offline access and local libraries and need to install and configure . And hands on suggestions.

In Web version it woks online and no local libraries required, chances of server mal-functions. Collaborative work offering.

**Network features reference**

Loading a webpage involves a number of Requests and Responses between the browser and the server

studio.code.org – play around algorithms

**API :**

**API** just a piece of software code to talk to another, example **REST API** is used to serve the client requests and server responses.

API that confirms to the **REST architectural style** is called REST API. **REST over HTTP**

**HTTP** is a protocol used to communicate through web server. it is a client-server protocol,

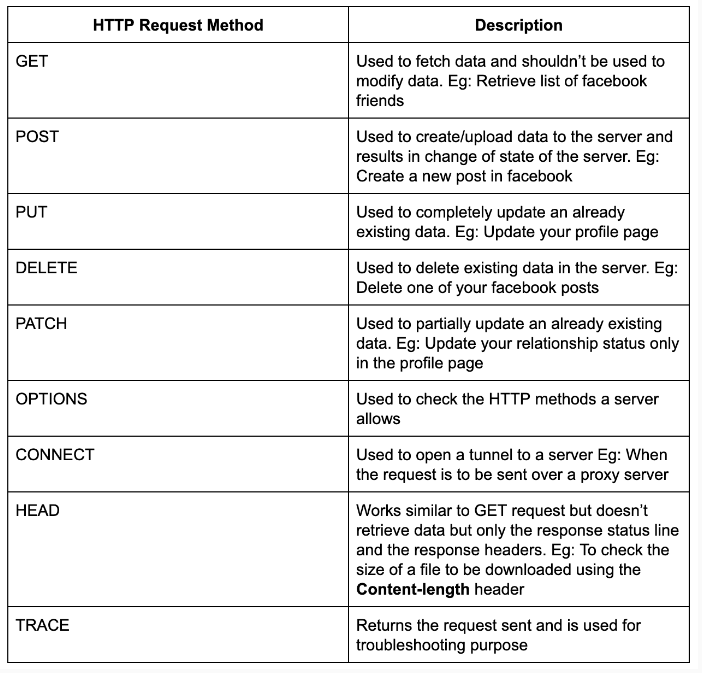
**HTTPs** is set of rules used to communicate securely.

**Request Method: GET, PUT, POST and DELETE**

**DOM is tree-like a structure, which the browser creates HTML content receives from it.**

Developer Tools provides a set of views for Frontend Developers to utilise

Networks tab allows to inspect network requests made by web applications

****

**Status code: 200 ok, 500 series is server-side errors 400 series for client side error**

**Status codes - 2xx**

The 2xx family of status codes or status codes 200-299 signifies the HTTP request was successfully received & understood by the server. We’ve been seeing the 200 status codes all the way until now. That’s what we get when the server returns some resource for our request.

**Status codes - 3xx**

3xx family of status codes denotes that further action must be taken to complete the HTTP request made.

**Status codes - 4xx**

Getting a 4xx status code tells us that there was an error in the HTTP request sent by the client - that would be the browser if we are visiting web pages.

here are a couple more HTTP status code families - 1xx & 5xx. 1xx is for information purposes while 5xx signifies there was a server error.

curl -V and postman to test http requests

**Content-Type: -- the tech used in current web site – ex: text/html**

Structure: a process used to build something

Structure Data - **JSON** - key and value a better way for computer tom understand.

To get html pages or we pages mostly preferred protocol is **HTTP** (Content-Type**: text/html**). Either side when web site needs a data from database then it better to go for **REST** due to it uses JSON structure (**Content-Type: application/JSON**) ex: posting comment.

* **Developer Tools provides a set of views for Frontend Developers to utilise**
* **Networks tab allows to inspect network requests made by web applications**
* **Elements tab allows you to inspect HTML and CSS of a web page**
* **This comes in handy to debug CSS issues as well**
* **Tools like the Box model view and Flexbox tool helps debugging easier**

REST APIs commonly use the HTTP protocol to send requests & receive responses.

It is in terms of the data returned that an API request differs from a usual HTTP request for a webpage.

* HTTP requests for webpages return HTML, CSS & JavaScript files which are rendered by the browser and displayed to the user.
* But, in the case of APIs, the request can be for any data (not just webpage) and the response is read by the requesting program which interprets the data.

JSON (JavaScript Object Notation - how cryptic :| ) is a standard data format that is easily “understandable” by applications

* It can be handled well in most languages
* So the data format in REST is usually JSON

Curl – client URL – CLI which works for non GUI client requests.

Copy a payload from network tab try POST method.

**Micro Experience:** developing code testing code and making production ready code

#### HTTP Caching and cookies

<https://developer.mozilla.org/en-US/docs/Web/HTTP>

The HTTP cache stores a response associated with a request and reuses the stored response for subsequent requests.

Types of Caches:

Private caches

Shared caches

Proxy caches

Cookies were once used for general client-side storage.

Cache-control:

Set-Cookie: or Cookie

**REST API calls using Programs**

* import java.io.BufferedReader;
* import java.io.IOException;
* import java.io.InputStreamReader;
* import java.net.HttpURLConnection;
* import java.net.MalformedURLException;
* import java.net.URL;
* class restAPI {
* public static void main(String[] args) throws MalformedURLException, IOException {
* // create url
* URL url = new URL("https://crio-xflix.herokuapp.com/v1/videos/602d228e672f010020e5e95d");
* // Send Get request and fetch data
* HttpURLConnection conn = (HttpURLConnection) url.openConnection();
* conn.setRequestMethod("GET");
* BufferedReader br = new BufferedReader(new InputStreamReader(
* (conn.getInputStream())));
* // Read data line-by-line from buffer & print it out
* String output;
* while ((output = br.readLine()) != null) {
* System.out.println(output);
* }
* conn.disconnect();
* }
* }

**Data Structure:[DSA]**

This is terminology/ formulae to store, retrieve and use the data effectively.

**Algorithm :**

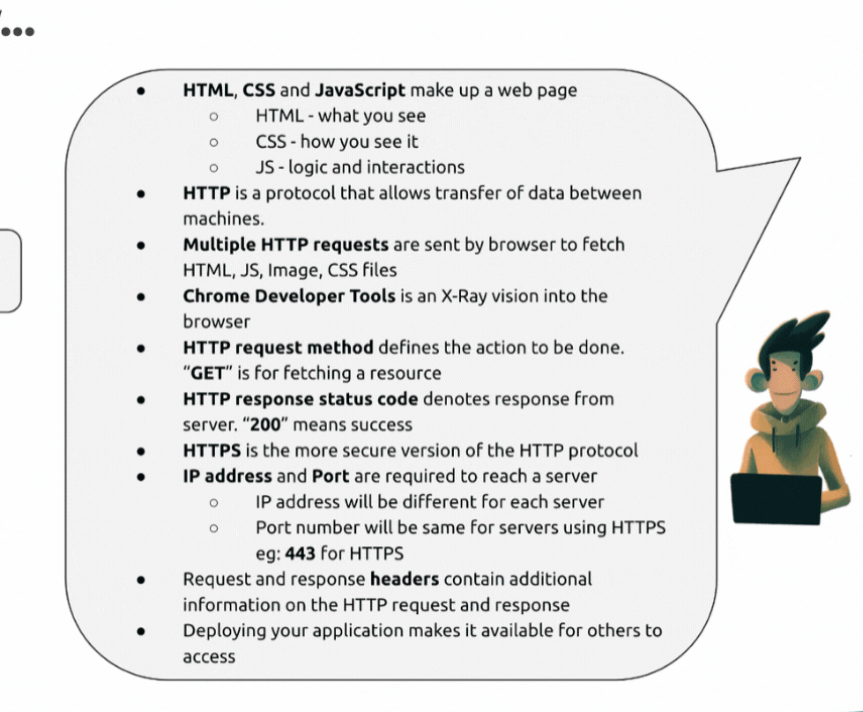
Is a set of instructions to perform a task to given to given to computer via program based on certain requirements.

**Pattern:**

Two pointer – using two variables – time complexity: O(n) – search O(log n) – sort O(n log n)

Brute force – two nested loops – Time complexity: O(n\*n)

Sliding window technique – O(n)



**Deconstructing the web contains HTML + CSS+JS+IMAGES**

Browsers use HTTP requests to fetch us web pages.

When we enter a website URL, the browser creates an HTTP request for the HTML of the page, on our behalf, and sends it to the server on which the website is hosted.

Query parameter vs Path parameter

Query start with “?” after the end point to request – to retrieve specific ID

/users?id=123 # Fetch a user who has id of 123

and where Path parameter start with “/” after endpoint to navigate. – to retrieve what ever data is present at other side

/users/123 # Fetch a user who has id 123

 If you want to identify a resource, you should use Path Variable. But if you want to sort or filter items, then you should use query parameter.

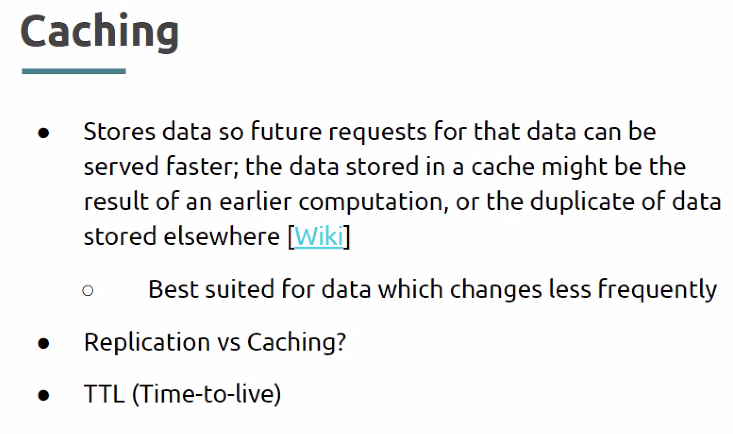
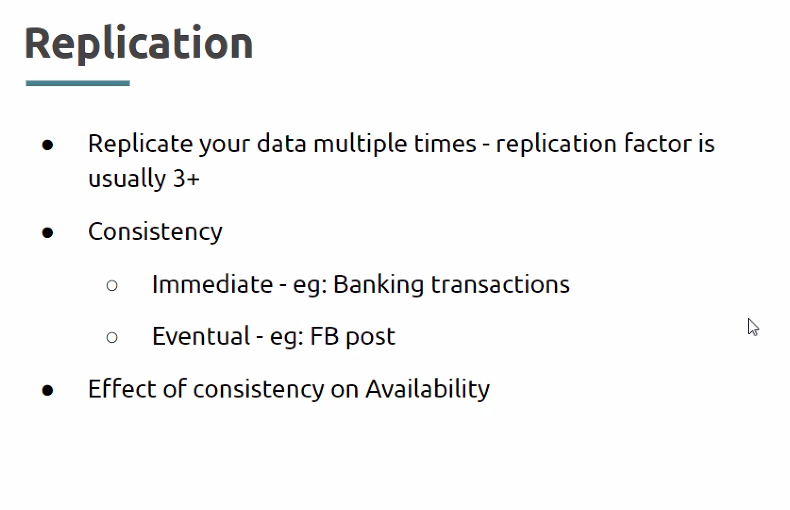
#### System Design

#### Distributed system:

1. Multiple computers2. Network Connectivity3.Work together for a common goal

Connecting multiple computers to shared network and working on same goal.

3-Tier Architecture > frontend (presentation layer) + backend (business logic) + (backend) Database



Load Balancer

# Date

const date = new Date({optional : date);

// Full date and time format

const fullDateTime = **date.toLocaleString**('en-IN', {

weekday: 'long',

year: 'numeric',

month: 'long',

day: 'numeric',

hour: 'numeric',

minute: 'numeric',

second: 'numeric',

hour12: true,

});

// Date only format

const dateOnly **= date.toLocaleDateString**('en-IN', {

weekday: 'long',

year: 'numeric',

month: 'long',

day: 'numeric',

});

// Time only format

const timeOnly **= date.toLocaleTimeString**('en-IN', {

hour: 'numeric',

minute: 'numeric',

second: 'numeric',

hour12: true,

});

// Short date format

const shortDate **= date.toLocaleDateString**('en-IN', {

year: 'numeric',

month: 'numeric',

day: 'numeric',

});

// Custom format

const customFormat **= date.toLocaleString**('en-IN', {

year: '2-digit',

month: '2-digit',

day: '2-digit',

hour: 'numeric',

minute: 'numeric',

});

console.log('Full Date and Time:', fullDateTime);

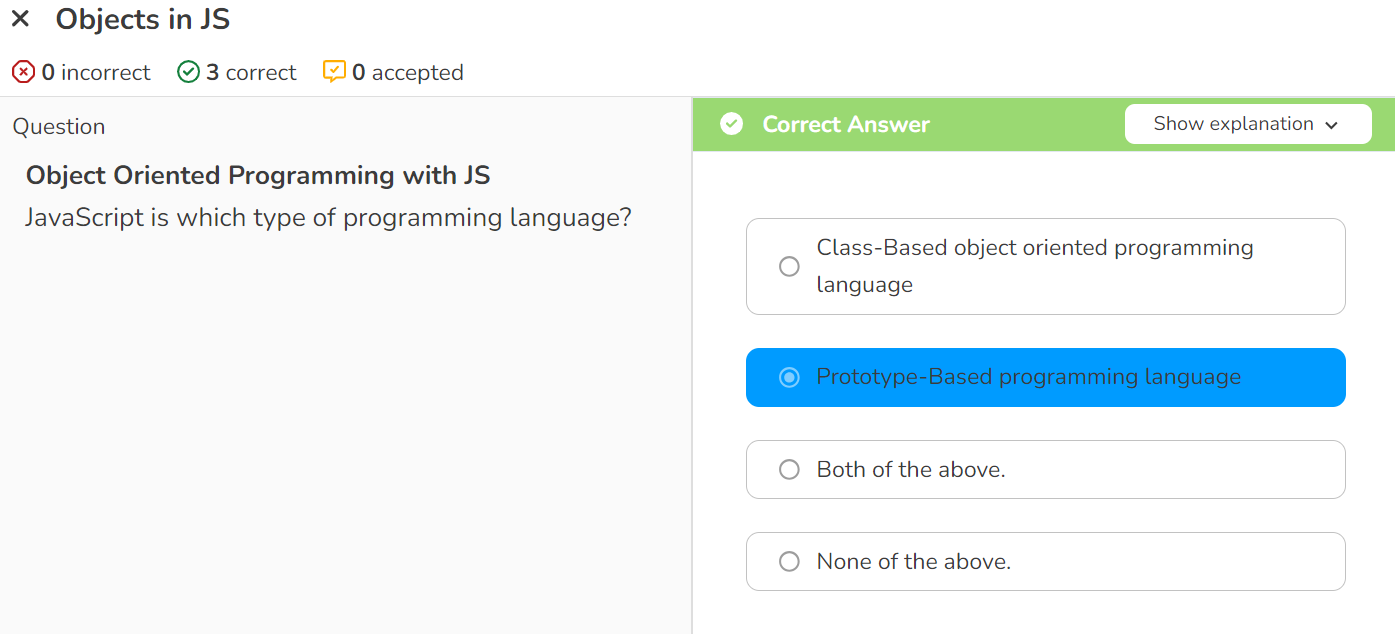
console.log('Date Only:', dateOnly);

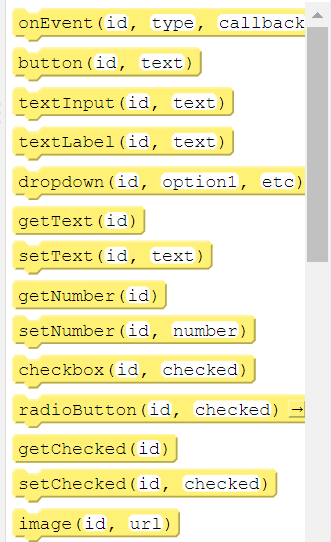
console.log('Time Only:', timeOnly);

console.log('Short Date:', shortDate);

console.log('Custom Format:', customFormat);

# **JAVA SCRIPT**





Local scope – function scope and block scope

Global scope -

Lexical scope - the scope which used by a child/ nested functions , the nested function can have access to variables declared in parent scope, lexical scope or lexical environment forms the closures.

#### All ABOUT VAR

* var variable can be redeclared and reinitialized.
* var global scoped. if it declared in function, it has function scope
* if var declared in block also can be accessed outside too.
* var allows variable hoisting, it causes possible errors.
* var declarations are either globally scoped or function-scoped

#### All ABOUT LET

* let can’t redeclared, it throws error, can be reinitialized.
* let have local scoped, If it is declared inside block , can’t accessed outside. Block scope
* let won’t support hoisting.
* let declarations are either block scope or function scoped

#### All ABOUT CONST

* const can’t redeclared and can’t reinitialized. Must assign at first declaration.
* const local scoped, If it is declared inside block , can’t accessed. Block scope.
* Local scope – block scope, function scope

#### All About primitive and non-primitive data types

* **Primitive data types are** number, Boolean, strings, null, undefined and NAN. Inbuilt classes.
* **Non - Primitive data types are** arrays, objects and functions. Even these data types declared with const can be modified. non-primitive data types reference to same memory even they assigned. Obj a = b;

typeof – is used to check the data types of variable

where in the case of checking type of null directly console.log(typeof(null)) => logs the type is object due to the null and objects coincidentally matches the same 32 to bit size, so returns the object

**NAN –** if a devision is invalid like 2/”a”

#### All ABOUT Arithmetical Operators

To compare the variable value with its data type a ===b or a !== b(Strict mode).

Truthy and falsy operators -if(a) returns true if it is value other wise false.

Where truthy values are used as condition returns Boolean value of true, inca of falsy vaues evaluates to false.

#### All About Strings

immuatable

Template Strings `I am template string ${varibale}`

Str.length;

Str.indexOf()

Str.substring(start index, end index)k

Str.substr(startindex, length )

Str.split(“”)

Str.slice(from, to)

#### All About Arrays

Arrays are objects, hetrogeonous, index access

Two find a variable is array or not use Array.isArray(var)

Arr.length, arr.push(), arr.unshift(), arr.pop(), arr.shift(), arr.slice(), ar.splice(at, no.of, add ele..)

Arr.reverse(), Arr.joins(“”)

Arr.sort() – it works for string array

For number array – Arr.sort((x,y) => x-y))

Remember in shift operation the empty space is removed once delete is accrued in array, elements shfited to left

Where unshift create a empty space and adds the element

#### All About Objects

Objects are non primitive data types

Object properties are retrieved using dot or bracket

Object.name = “snb” or Object[“name”] = “snb”to add

Object [1] or Object[“1”]

Delete Object.age

#### All About De-structuring and spread operator

A process to retrieve the object or arrays values to targeted variable

const [ index0, , index2] = arr;

for arrays must declare equal number of array elements

const {name , age} = obj; // when obj properties have same name

**spread operator**

const arr1 = {…arr} -> to copy the array to another array

const obj1 = {…obj}

shallow copy - assigning the one object/var to another object/var, in non primitive data types both objects refers to same reference. So any update in values both types gets update

deep copy – where it create a new copy , example: spread operator

Shallow copying creates a new object with references to the same memory locations as the original object, while deep copying creates a new object with new memory locations for all of its properties and nested objects or arrays.

#### All About Math Functions

Math.floor() – to round of the integer to smaller integer side

Math.ceil() – to round of the integer to larger side

Math.round() – it rounds the number based on >=0,5 or < 0.5

Math.abs() – to return positive integer

Math.trunc() – to cut the integer to non decimal value.

Math.sign() - returns the -1 or 1 based on negative or positive number

Math.max() – to return the max value

Math.min() – to return the min value

To retrieve constants

Math.PI and Math.E

#### All About pass by reference and pass by value

**Pass by reference** is achieved by non primitive data types, where these data types are refers to same memory location even when modified or assigned to another variable.

**Pass by value** is achieved by primitive data types, because every new assign will create new reference in memory.

#### All About Arrow functions

* Must use the () – open and closing brackets , If a arrow function don’t have parameter. If a single argument no need to use brackets.
* More than one argument must use brackets (a,b)=>{}
* If a function have only one statement , can return it without a return keyword

#### All About Hoisting, Closure and Callback

**Hoisting** special behaviour of var keyword, even using of a variable can works when it declared after that. It called hoisting

**Closures** is a process of returning a function inside another function without calling, and assigning it to a variabale and calling the variable can do the same process. It is achievable because of lexical environment scope. The nested function lies in lexical environment so it forms the Closure

**Callback**  is a process to handling the function call to another function. Where a function is passed a an argument to function, and calling that callback function inside it at target place.

#### All About Array Advance Method

The advance array methods are callback function are used to work with arrays . where arrow function must be implemented for every callback function,

**forEach((element)) –** used to iterate through the array.

**filter((element)) -**used to return the elements based on filter condition

**find((element)) –** used to return the first occurrence of element based on find condition

**sort((element1, element2))** – used to sort the array, default sorting order is lexicographical(dictionary) order best suits for strings, for number array pass the comparator function to sort the array. (a-b) ascending order, (b-a) is used to sort in descending order.

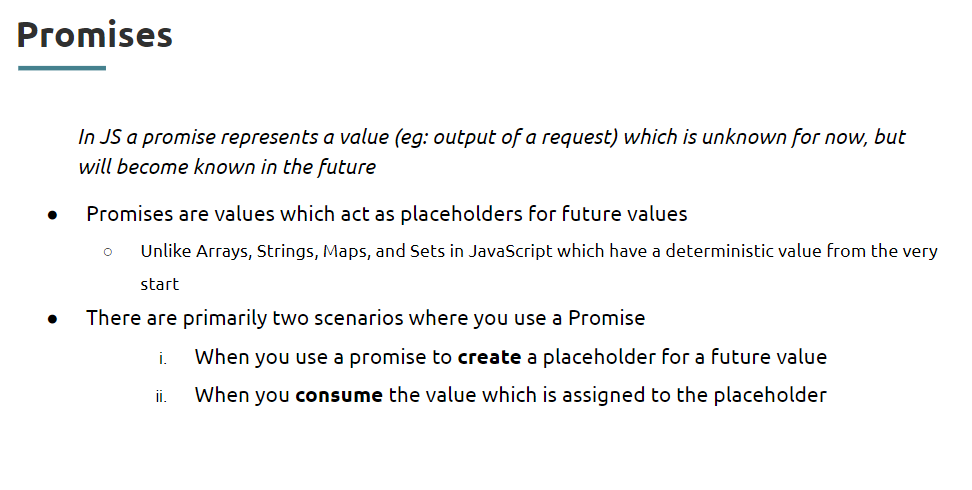
**map((element))** – used to return the array modified array.

**reduce((a,b)=>{a+=b},0) –** returns thereduced value of the array elements based on the reduce condition.

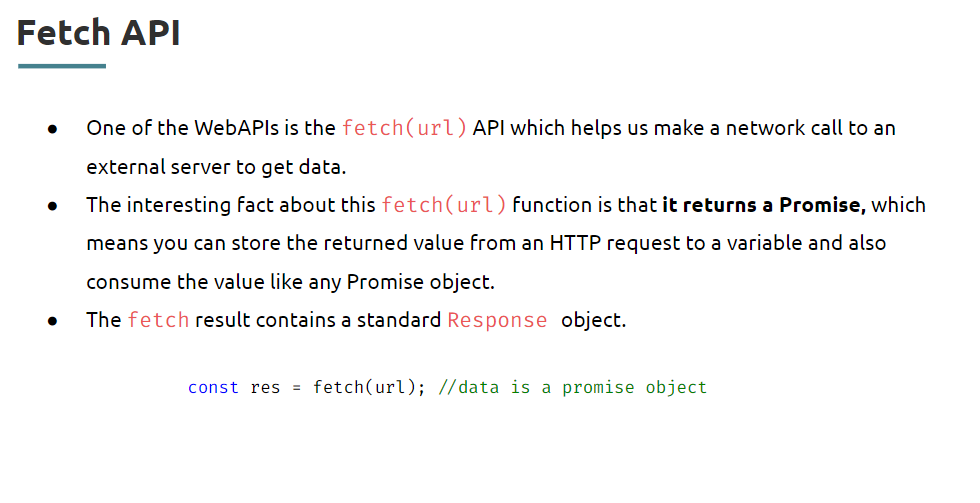
#### All About Promises

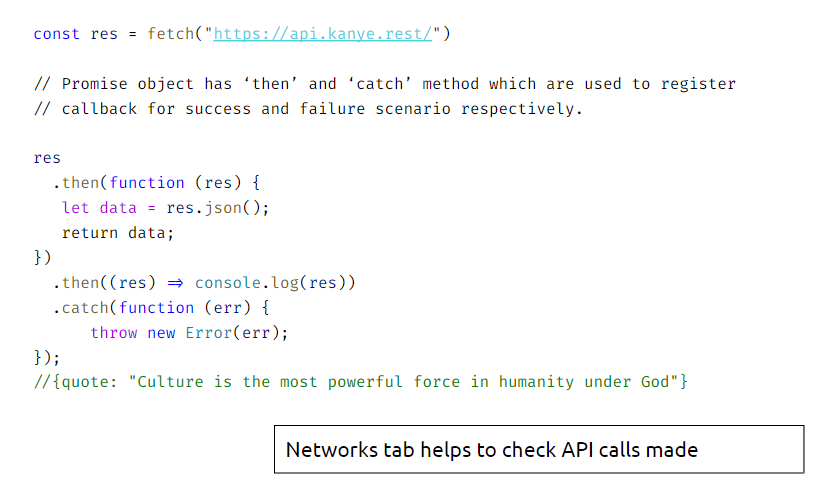
Asynchronous a way command the browser to execute a desired function (parallel task) where promises takes cares the process must execute and assures a output.

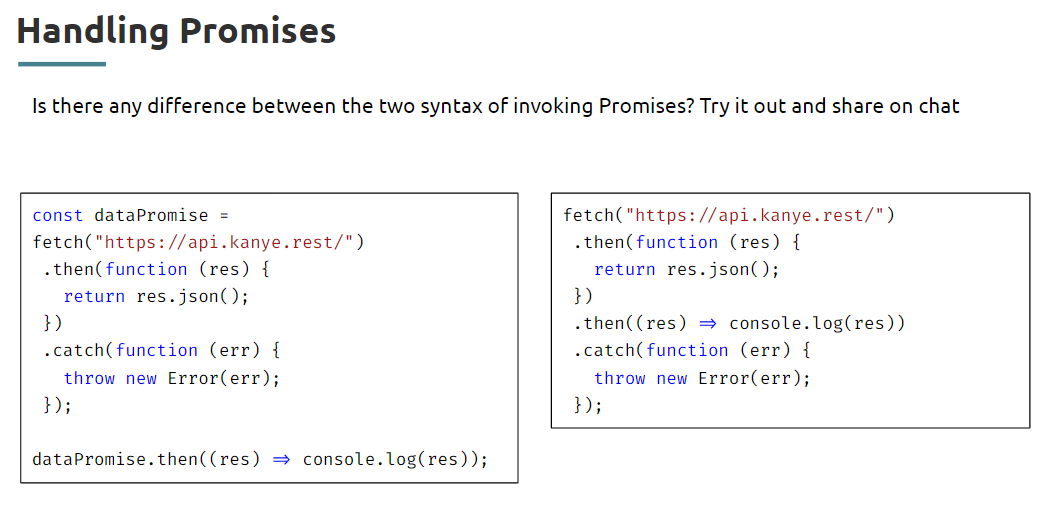
Asynchronous – not necessary for a operation to be completed for another to be get started. Where Asynchronous tasks are handled and executed by browser and where normal code taken care by Java script. Promise is an object



Async and Await are the generators, which pass the execution of js code until it returns a value. converts asynchronous code to synchronous code.

Fetch API:  


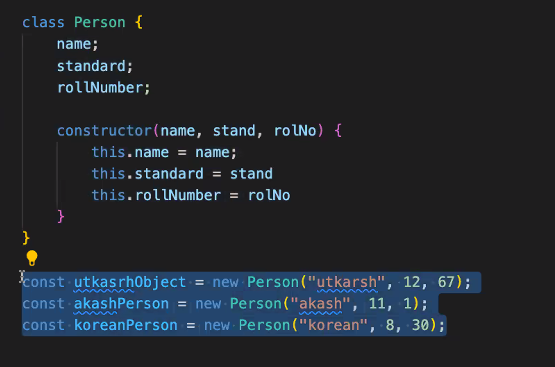




Fetch API returns a promise, When a promise is resolved using .then can process it further to get the data . to handle the reject state can use the .catch to handle it.

#### All About Class & Objects

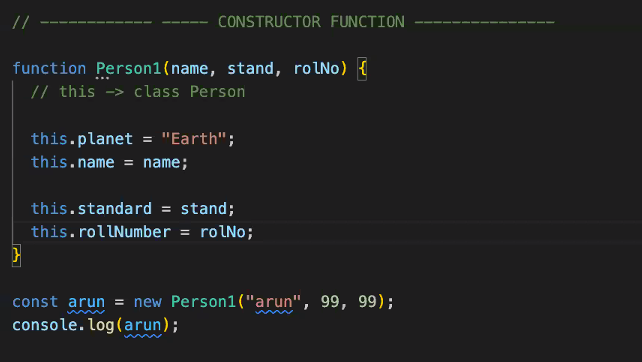
<https://github.com/thakurutkarsh22/FE-3-March-2024>



“this” is for current and “super” for parent

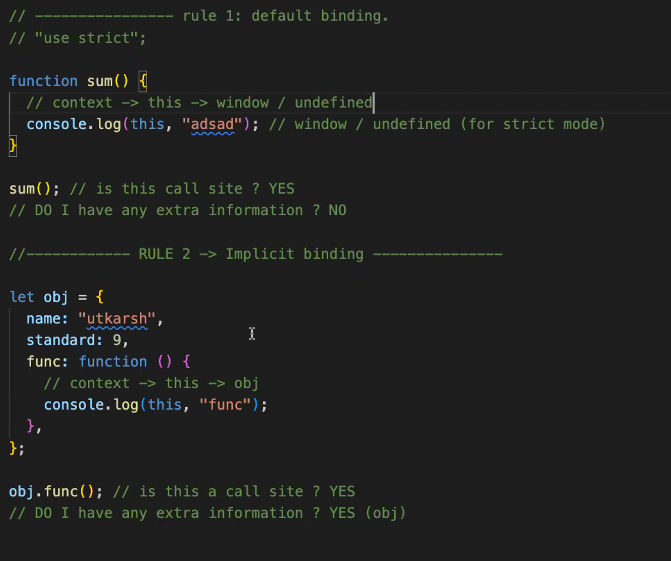
#### Functional Programming:

Array is an object , function are nothing but 1st class



The objects of functions can be same if constructed by based on class,

the first object for JavaScript engine is window.



Striver

Kunal waha

Arpit biyani

Sumit malik

Javascript.info, you don’t know js

# **HTML && CSS**



**<!DOCTYPE> this tag is used to let know the browser to the tyf document and html 5**

**<head and title>**

**<heading tags are from h1 to h6 these are differentiate the size of heading>**

**<paragraph tag is used to implement the para text>**

**<ordered list and un-ordered list >**

To style the ordered list use **type** attribute and to style the unordered list use style attribute with value **list-style-type.**

**Ol style**

type="1" The list items will be numbered with numbers (default)

type="A" The list items will be numbered with uppercase letters

type="a" The list items will be numbered with lowercase letters

type="I" The list items will be numbered with uppercase roman numbers

type="i" The list items will be numbered with lowercase roman numbers

**Control List Counting**

By default, an ordered list will start counting from 1. If you want to start counting from a specified number, you can use the start attribute.

**ul style**

disc Sets the list item marker to a bullet (default)

circle Sets the list item marker to a circle

square Sets the list item marker to a square

none The list items will not be marked

**<anchor tag and href> defines a hyperlink, hrefy indicates the destination**

**The content for anchor tag can be image or text**

**Relative links:**

Can route to internal html file s

**Absolute links: the address of an IP example** [**www.html.com**](http://www.html.com)

**Link bookmarks:** To link internal tags of a html page use hash symbol # plus the value of the id attribute for the element that you want to internally(with in the file) link to, usually further down the page. You then need to add the same id attribute to the element you are linking to.

# and dead link

**Attribute Values**

**Value Description**

\_blank Opens the linked document in a new window or tab

\_self Opens the linked document in the same frame as it was clicked (this is default)

\_parent Opens the linked document in the parent frame

\_top Opens the linked document in the full body of the window

Framename Opens the linked document in the named iframe

Use <bold tag> to bold the text

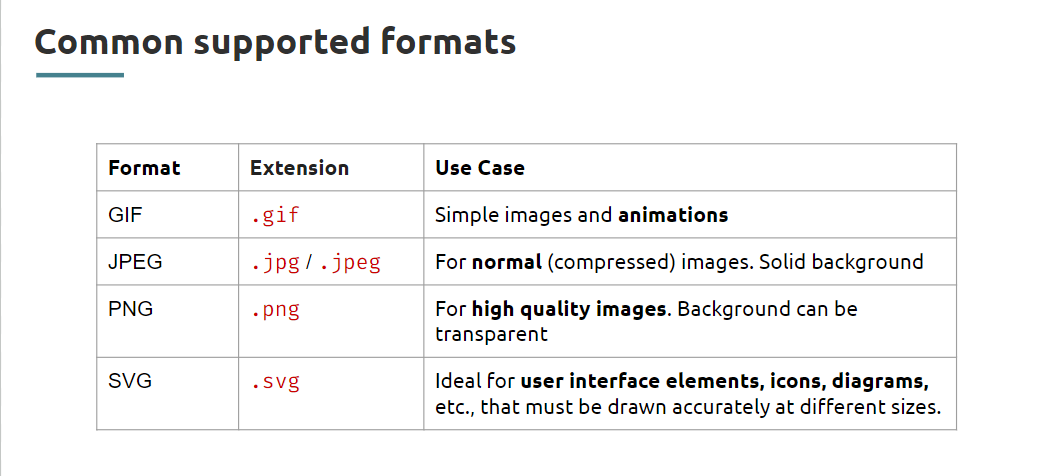
Target attribute with value \_blank opens the url in new tab.

**Image tag, break tag, horizontal rule<hr>– self closing tab <img> or <hr/>**

**Required attributes**

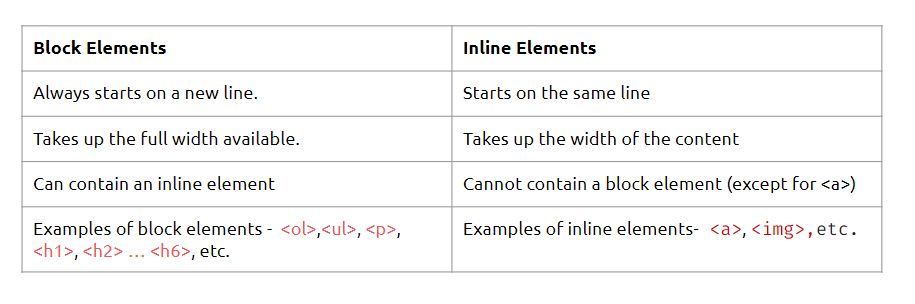
* **src** – to provide the source path of the Image
* **alt** - alternative text when image not available

**optional height and width**

****

**Block elements – headings, lists, paragraph**

**Inline elements – images, links**

****

**Inline elements** automatically adjust their **width to fit the content** they contain, taking up as much width as necessary, without forcing line breaks or expanding to fill the available horizontal space. it occupies the required space in the container.

On the other hand,**block elements** occupy the **entire available width horizontally** by default. It occupies entire space of a container.

**<div tag> act as a container**

To divide the page and style it as per requirement – accepts the attribute and with their property and value.

**<span tag> to highlight the divided part of an paragraph**

**<i> - to make italic**

**<b> - to bold**

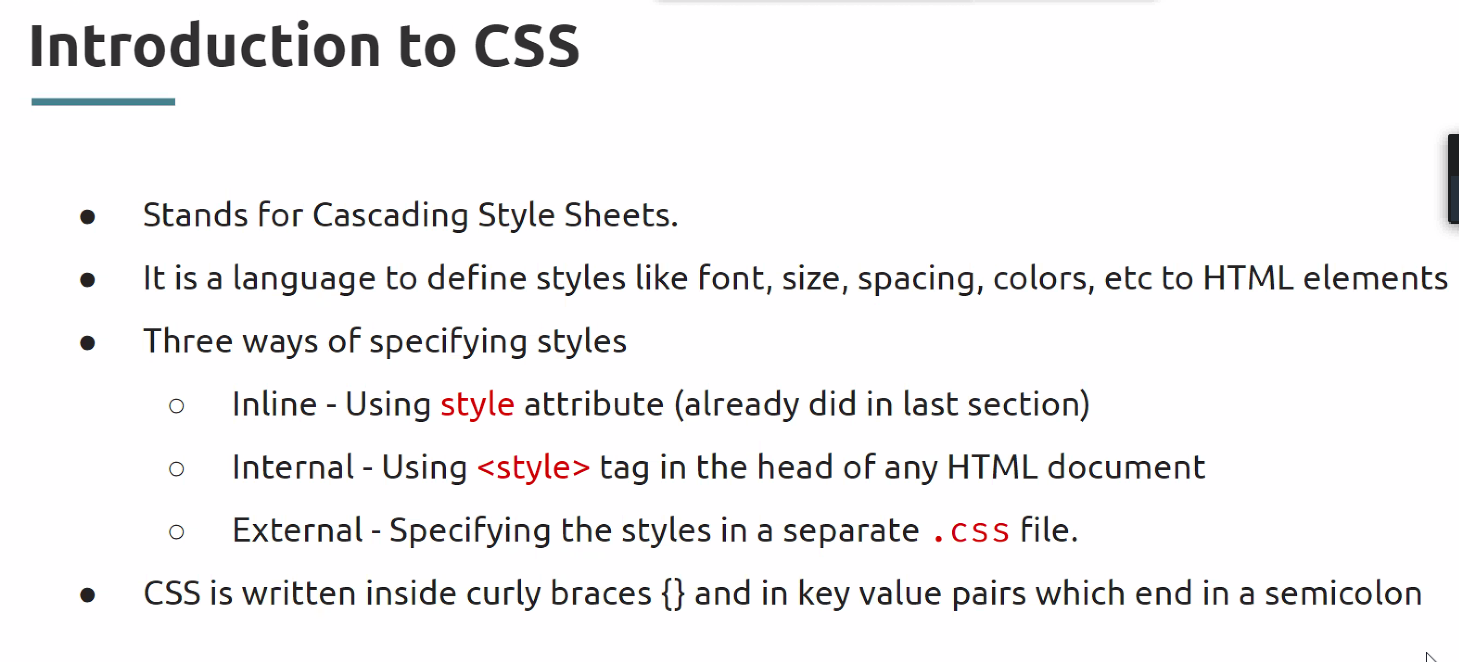
**<em> - to make special word in statement**

**Sematic and non-semantic tags**

div and span are non-sematic elements.

**Sematic tags** gives more info an the content. Tags are section, header, nav, footer, and article.

section is used for different topics and where article is used for specific group same types.



Animation, Transition and transform are important properties for css.

**Selectors in CSS** used to choose style the specific tag/element without any prefix, for the class attribute selector need to use dot prefix. And for the id selectors use # has the prefix

**Sudo selectors**

– the hover added to the specially case

-- the active used for the when a link clicked

--visited

/\* unvisited link \*/  
a:link {  
  color: red;  
}  
  
/\* visited link \*/  
a:visited {  
  color: green;  
}  
  
/\* mouse over link \*/  
a:hover {  
  color: hotpink;  
}  
  
/\* selected link \*/  
a:active {  
  color: blue;  
}

Cursor properties in CSS:

The cursor property in CSS controls what the mouse cursor will look like when it is located over the element in which this property is set.

.auto { cursor: auto; }

.default { cursor: default; }

.none { cursor: none; }

.context-menu { cursor: context-menu; }

.help { cursor: help; }

.pointer { cursor: pointer; }

.progress { cursor: progress; }

.wait { cursor: wait; }

.cell { cursor: cell; }

.crosshair { cursor: crosshair; }

.text { cursor: text; }

.vertical-text { cursor: vertical-text; }

.alias { cursor: alias; }

.copy { cursor: copy; }

.move { cursor: move; }

.no-drop { cursor: no-drop; }

.not-allowed { cursor: not-allowed; }

.all-scroll { cursor: all-scroll; }

.col-resize { cursor: col-resize; }

.row-resize { cursor: row-resize; }

.n-resize { cursor: n-resize; }

.e-resize { cursor: e-resize; }

.s-resize { cursor: s-resize; }

.w-resize { cursor: w-resize; }

.ns-resize { cursor: ns-resize; }

.ew-resize { cursor: ew-resize; }

.ne-resize { cursor: ne-resize; }

.nw-resize { cursor: nw-resize; }

.se-resize { cursor: se-resize; }

.sw-resize { cursor: sw-resize; }

.nesw-resize { cursor: nesw-resize; }

.nwse-resize { cursor: nwse-resize; }

**Descendant selectors** are used when same style is need to update for the children elements, like .main h1,h2{declarations}

The :last-child selector allows you to target the last element directly inside its containing element.

**Priority order in selectors:**

* !important
* inline
* id
* class
* element

**Selector grouping** is way to declaring the all same style within same css definitions.

**Nth Child**

ul li:nth-child(even) { background-color: lightgray;}

p:nth-child(2) { color: blue;}

In normal, you can check in developer tools of a web page and observe some striked-off styles, conflicting styles for the same element from different selectors, the one with the highest selector specificity is used

**Inheritance in selector**

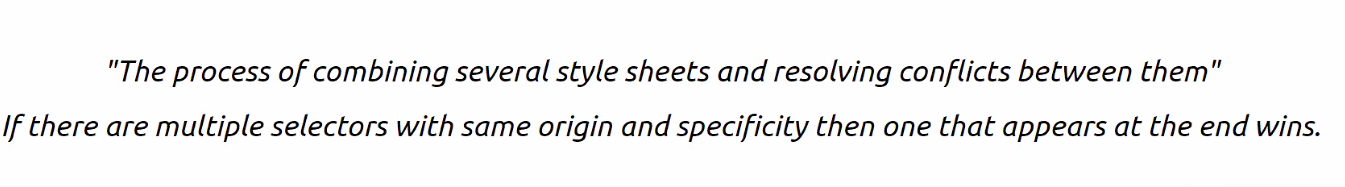
Inheritable properties in selectors:

Colors and fonts props– can be acquired to child tags

Non-inheritable selectors:  
size – can’t acquired to child tags

Width height

**Cascading in CSS**

****

**Colors:  
rgb(0/255, 0/255, 0/255)**

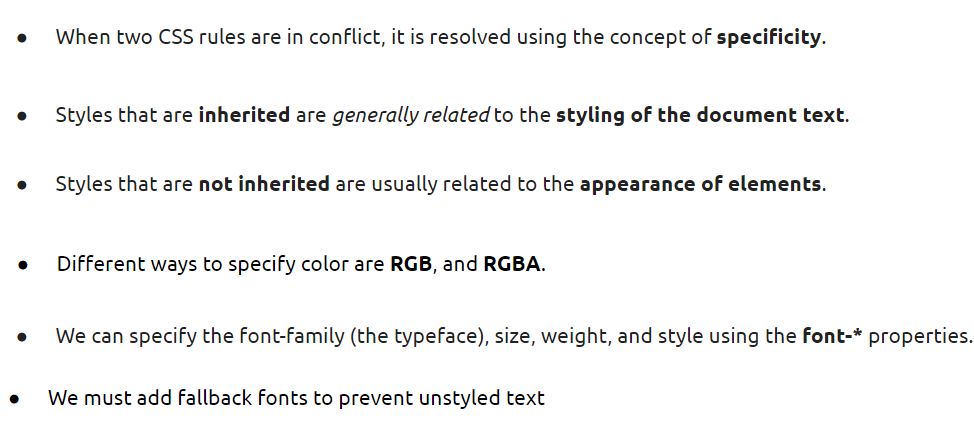
**rgba((0/255, 0/255, 0/255, opacity-0to1)**

**Fonts in CSS**

font-family - can give multiple font types for call backs if user device doesn’t supports defined style looks for next one to apply font style.

font-size

font-weight



**CSS box model:**

Padding & Margin:

Margin – top right bottom left

Padding is used to adjust the content area in the container, where margin is used to modifying to the element wise

Border – is used draw the outline

**Units in CSS**

* absolute
* relative

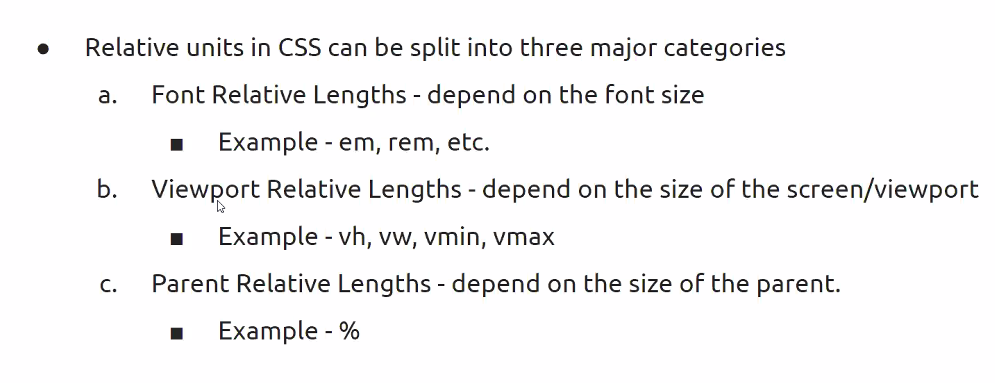
Absolute lengths:

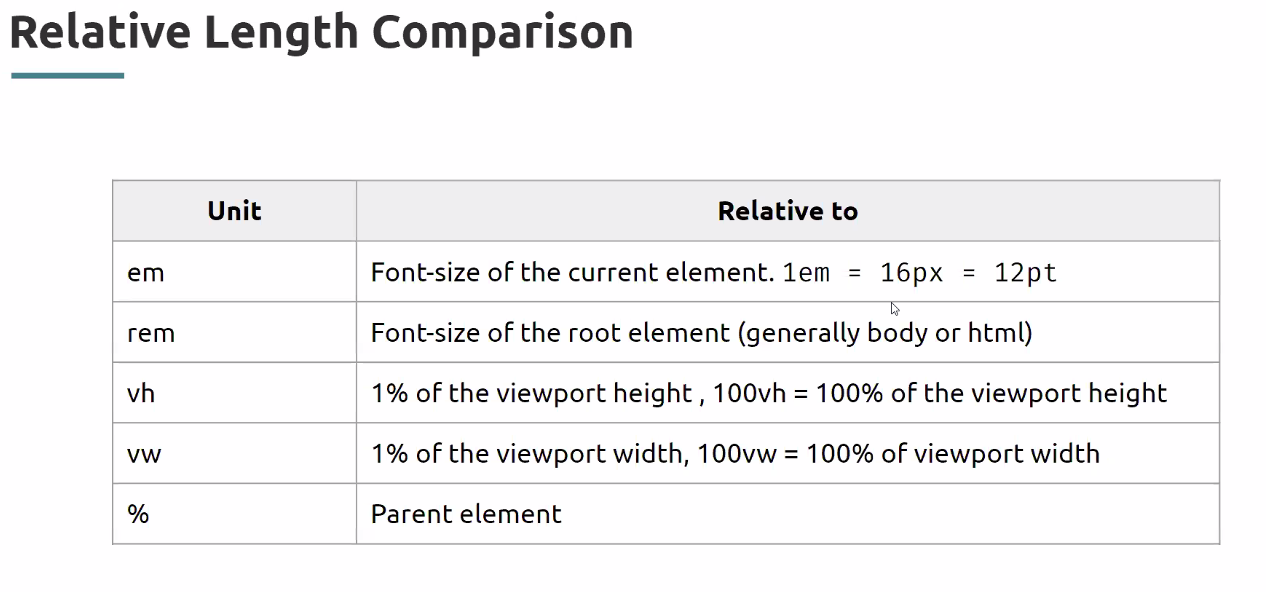
Absolute lengths displays the content same in every device.

Pixels,

Relative lengths:

rem and em depends on parent

vw and vh is 1% of the viewport  




The <aside> tag defines some content aside from the content it is placed in.

The aside content should be indirectly related to the surrounding content.

**Tip:** The <aside> content is often placed as a sidebar in a document.

aside {

width: 30%;

padding-left: 15px;

margin-left: 15px;

float: right;/\*is used for place the container at any position\*/

font-style: italic;

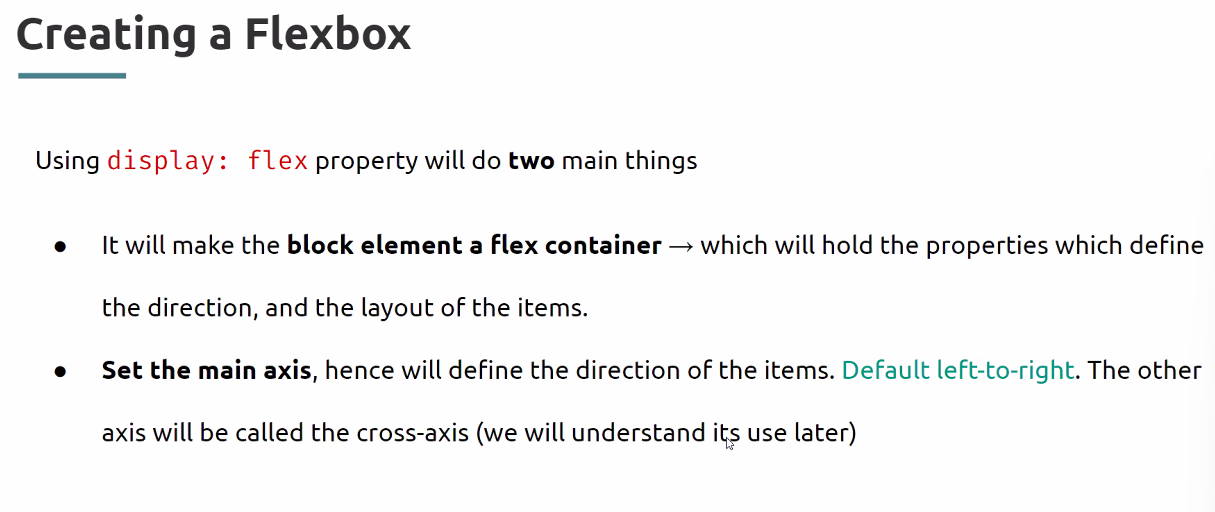
background-color: lightgray;

}

1. [**CSS: filter**](https://www.google.com/url?q=https://css-tricks.com/almanac/properties/f/filter/&sa=D&source=editors&ust=1644941300012574&usg=AOvVaw1PzW5RbnEc8XifdzxEyWn0)
2. [**CSS: transform**](https://www.google.com/url?q=https://css-tricks.com/almanac/properties/t/transform/&sa=D&source=editors&ust=1644941300013088&usg=AOvVaw2iCry_vho8t-PT5RvvavTw)
3. [**CSS: transition**](https://www.google.com/url?q=https://css-tricks.com/almanac/properties/t/transition/&sa=D&source=editors&ust=1644941300013497&usg=AOvVaw3m9EdBSoGiarmwPfuDjBpT)
4. [**CSS: overflow**](https://www.google.com/url?q=https://css-tricks.com/almanac/properties/o/overflow/&sa=D&source=editors&ust=1644941300013822&usg=AOvVaw2AucJv-xwyMLy_OsOayTLS)

## **Flex box in CSS**

Main and cross axis



justify-content: flex-start, center, flex-end, space-between

if there is no enough space between the list of items for flex property element, specify the space then do justify-content: space-between

align-items is used to adjust the content on cross(y) axis, where justify-content is used to adjust the items on main axis

### display: flex;

is used to enable the flexibility to the container.

.flex-container {  
  display: flex;  
}

### **Flex-direction:**

The flex-direction property defines in which direction the container wants to stack the flex items.

* column
* column-reverse
* row
* row-reverse

### **Flex-wrap:**

The flex-wrap property specifies whether the flex items should wrap or not.

* nowrap
* wrap-reverse
* wrap

### **Flex-flow**

The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.

* row wrap

### **justify-content**

to modify the content on main axis(x), The justify-content property is used to align the flex items:

* Center
* flex-start
* flex-end
* space-ground- to occupy entire container with space between items
* space-between

### **align-items**

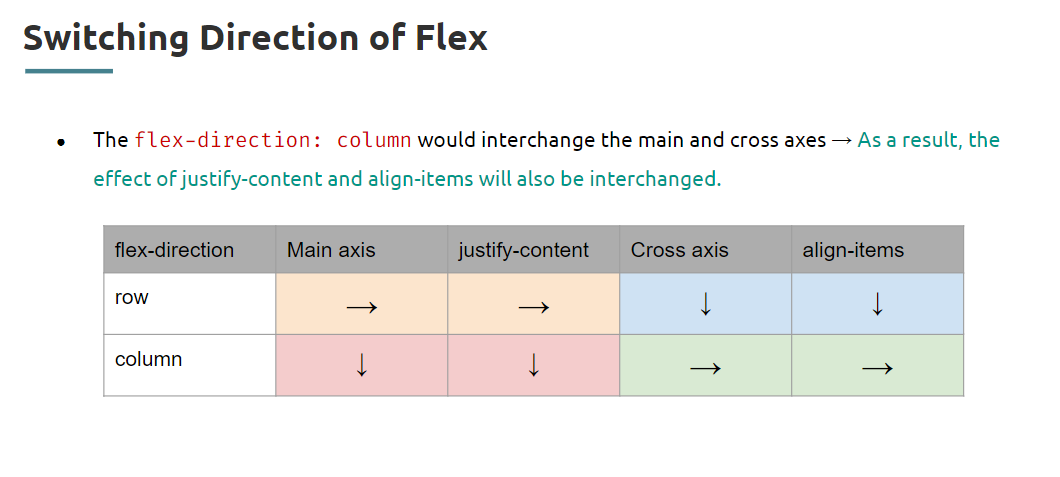
The align-items property is used to align the flex items. To modify the items on cross axis (y)

* Center
* flex-start
* flex-end
* stretch - The "align-items: stretch;" stretches the flex items to fill the container (this is default)
* baseline

### **align-content**

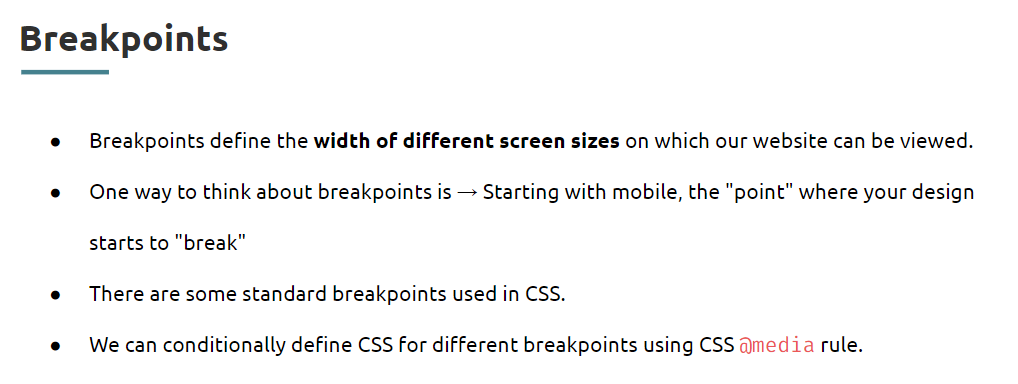
The align-content property is used to align the flex lines. Without any extra space between items – on cross axis modification , where multiple lines items presents then choosing align-content is good, it is combination of justify content and align-items.

* space-between
* space-around
* stretch -default
* center
* flex-start
* flex-end



## Responsive Design:

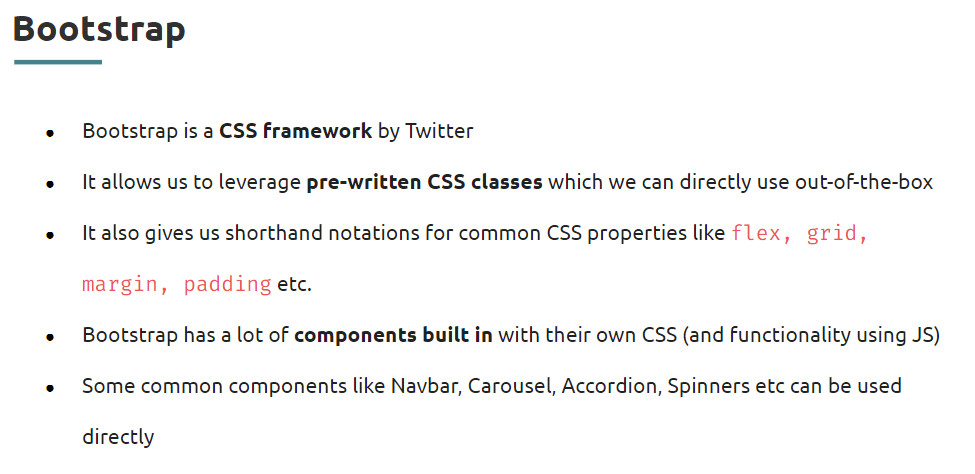
To build responsive web pages most used units are viewport units(vw,vh)



@media query(){styles}

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”/> - defining about the viewport to use @media

## **Bootstrap**



Link the required bootstrap for style in session head tag

This ensures the Bootstrap rules get prioritized if there’s a conflict between the styles in Bootstrap CSS and the CSS file you might have created.

Get started with the Bootstrap CSS library

* By adding the <link> element provided in the head tag, your browser plugs in the Bootstrap CSS file to your web page.

### Bootstrap alerts:

* .alert, .alert-primary, .alert-success classes are used to style the alert message.

### Bootstrap container:

* .container class adds the pad and left and right margin

### Bootstrap navigation bar:

* .navbar, navbar-brand, navbar-nav, nav-item, nav-link

### Bootstrap Accordion:

* .accordion-collapse, {flush},{borderless}

For collapse div

### Bootstrap car:

* Class = carousel
* carousel-inner, carousel-item
* carousel-control-prev, carousel-control-next

> **navbar** class is used for nav tag to define, where **navbar-expand** –{-sm|-md|-lg|-xl|-xxl} are used to specify when to the nav bar needs to expand

> **navbar- brand** class used to specify the brand name on nav bar

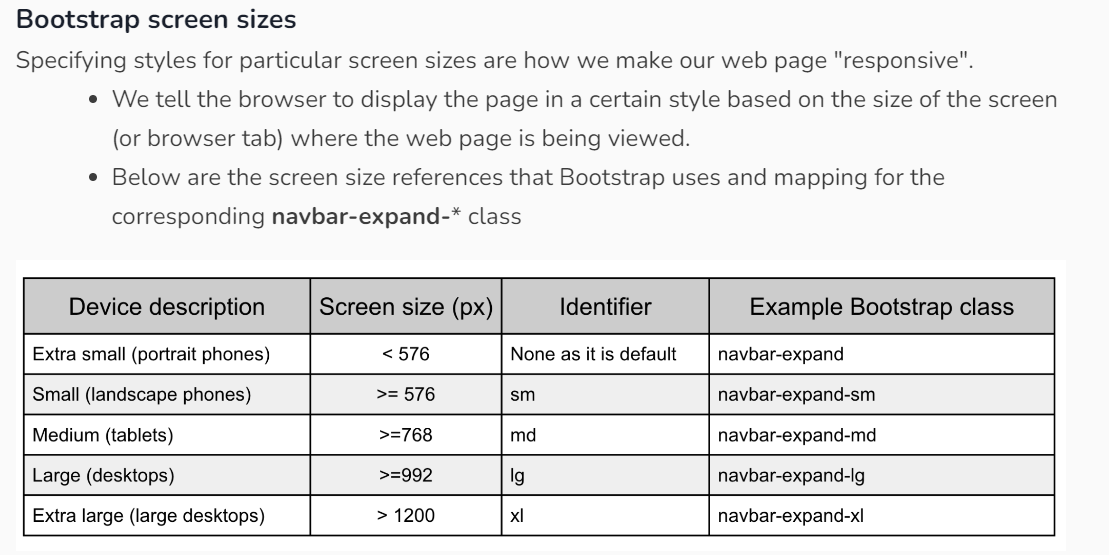
> **navbar-toggler** class is to specify the toggler button with span class **navbar-toggler-icon**

> to collapse button menu use script methods specify the target div id in button tag data-**bs-toggle="collapse" data-bs-target="#navbarSupportedContent"**

> use class **navbar-nav** for navigation items

### Bootstrap screen sizes:

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no"> is used to define the viewport definition to index html file.



### Breakpoints:

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap.

<https://getbootstrap.com/docs/5.0/layout/breakpoints/>

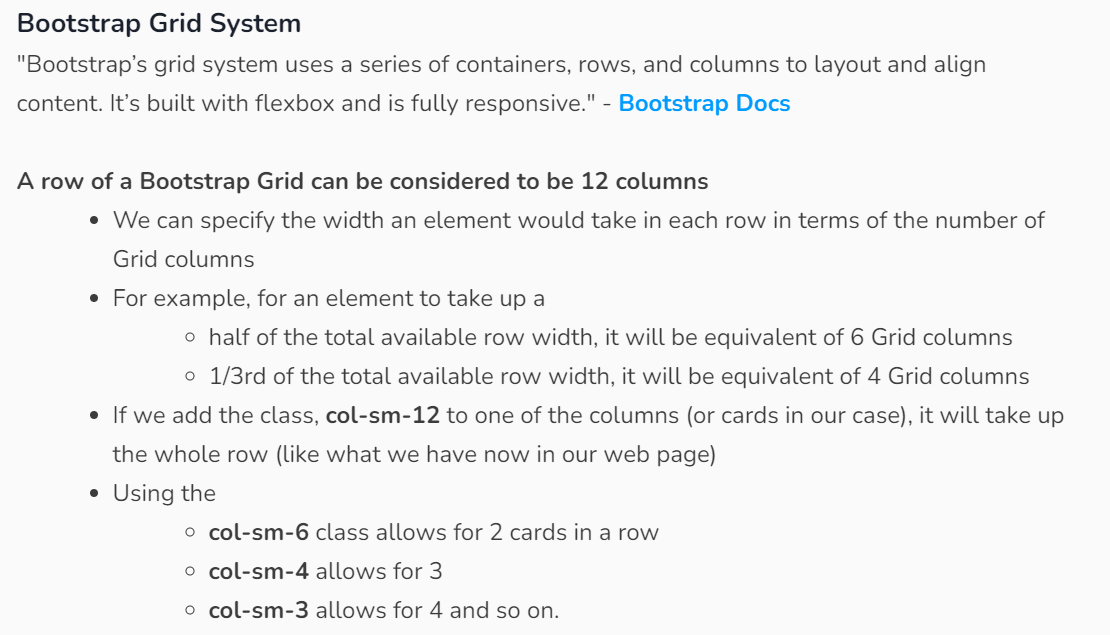
Toggle Icon and collapse in Bootstrap:

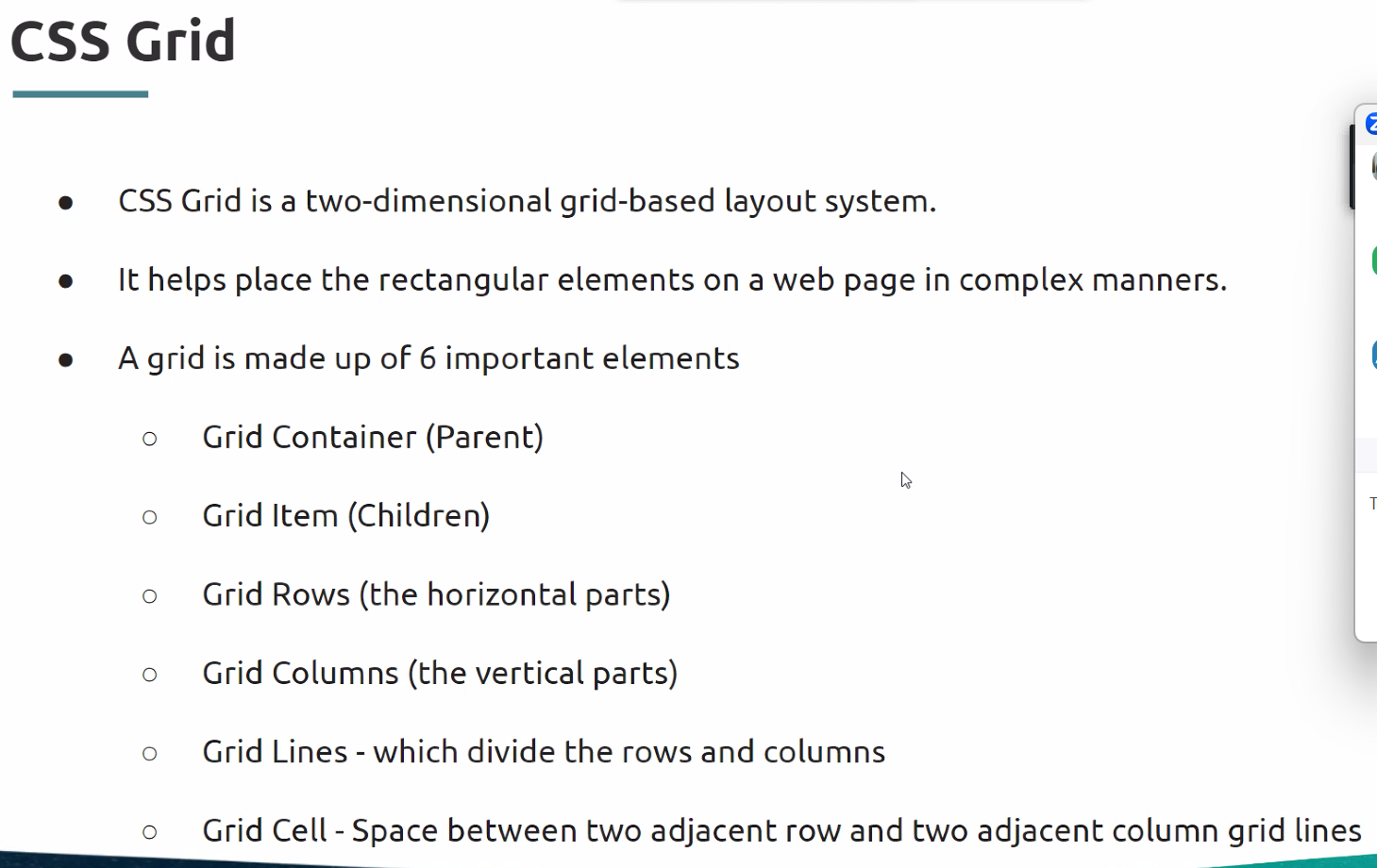
Collapse and hide activities :



<https://getbootstrap.com/docs/5.0/components/collapse/>

### Grid System:





<div class="container">

<div style="background-color: yellow">1</div>

<div style="background-color: green">2</div>

<div style="background-color: red">3</div>

<div style="background-color: blue">4</div>

</div>

.container {

display: grid;

grid-template-rows: 4rem 4rem 4rem;

grid-template-columns: 16rem 16rem 16rem;

}

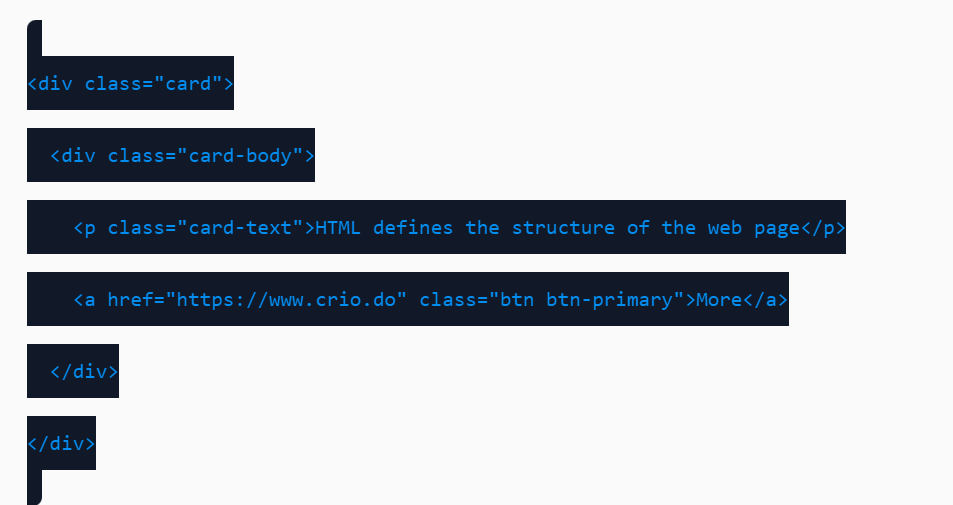
<https://codesandbox.io/p/sandbox/awesome-neumann-g9q2dg>

<https://getbootstrap.com/docs/5.0/layout/grid/>

a div with a row

a div with col

### Cards:



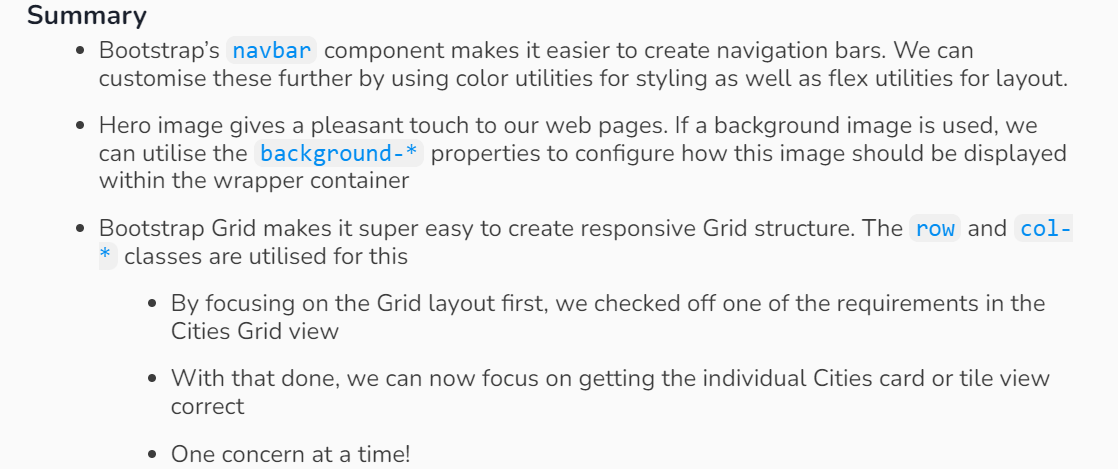
A div with card

A dive with card body

A div with card-title

A class with card-text

A class for card-footer

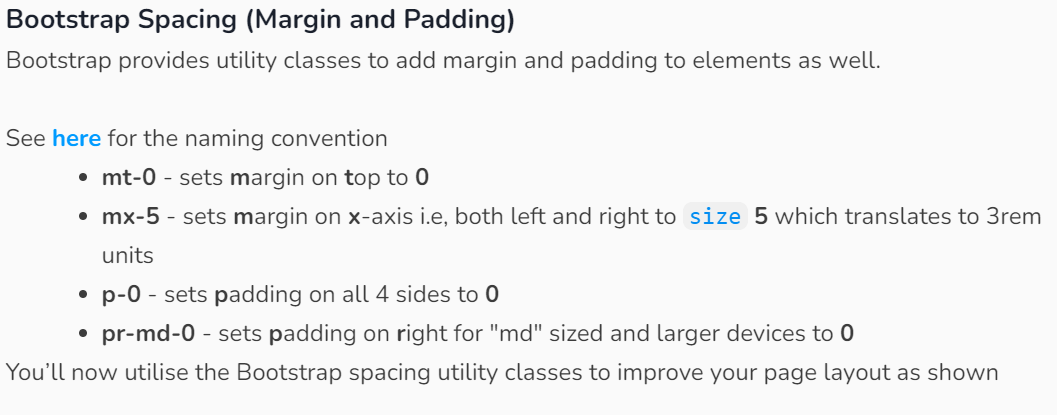


<https://getbootstrap.com/docs/5.0/components/card/>

buttons:

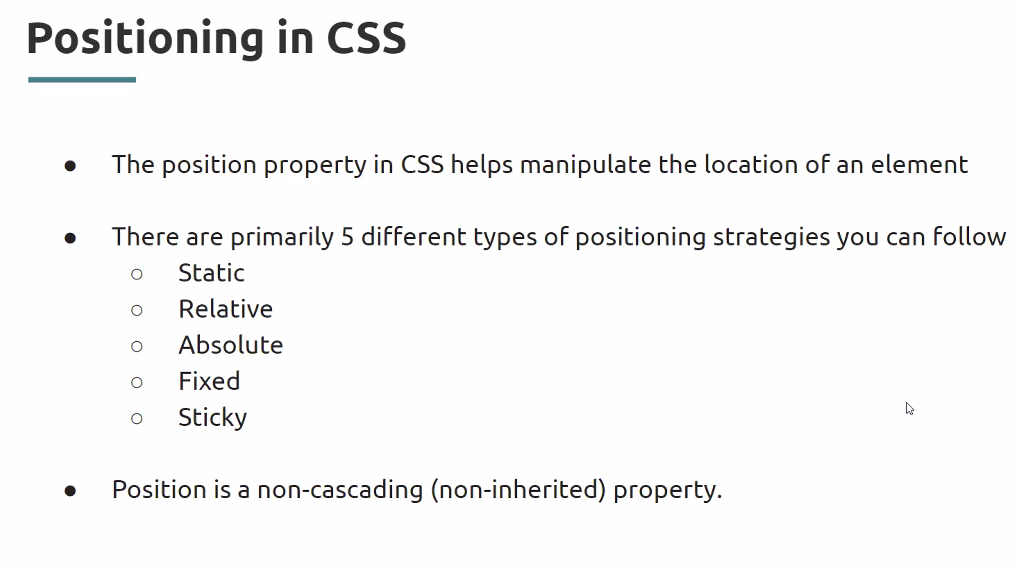
<https://getbootstrap.com/docs/5.0/components/buttons/>

### Bootstrap Spacing (Margin and Padding):

****

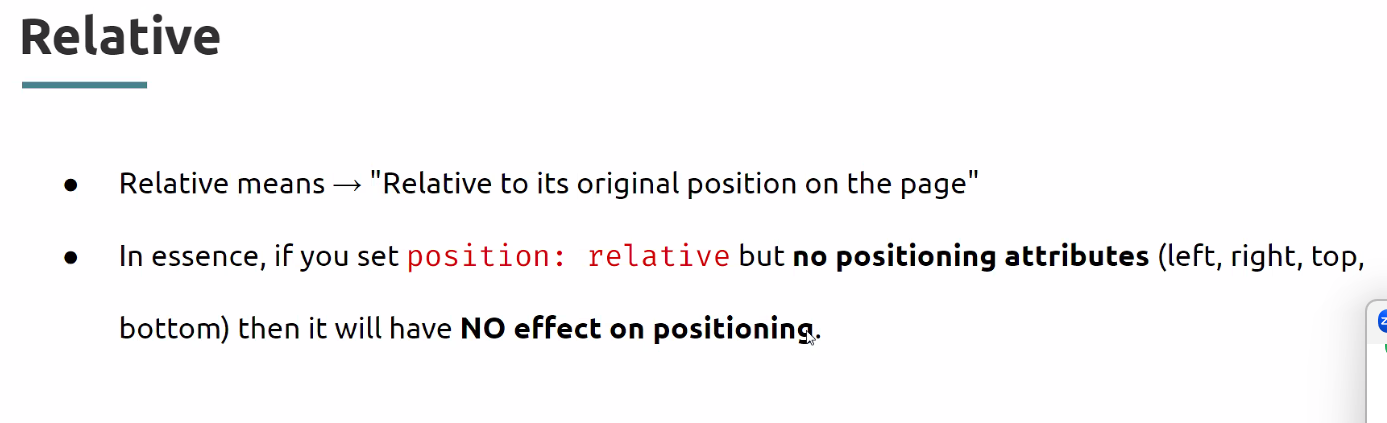
**task: combining the both nav bar module and header module**

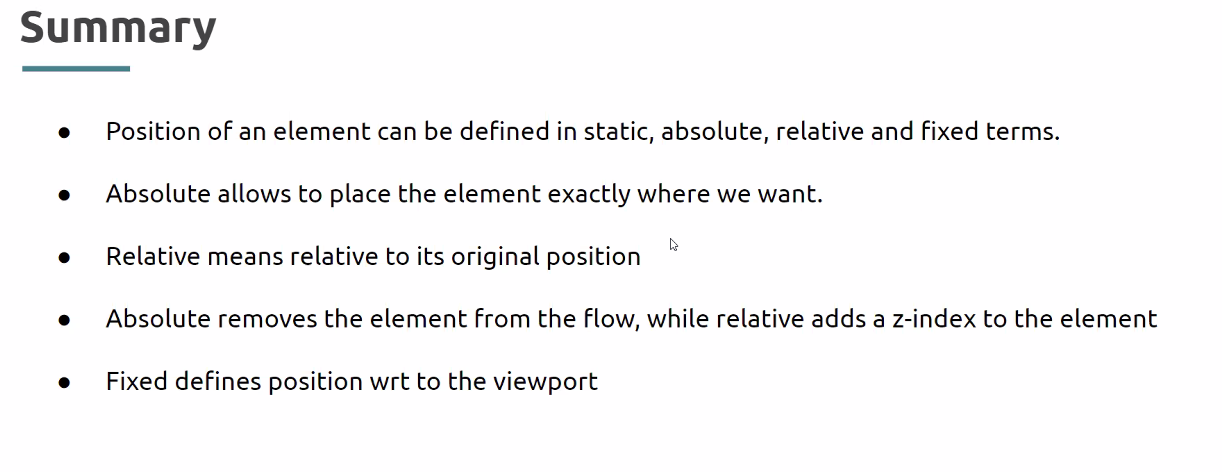
## **Positioning in CSS**



Static is default position , no change with any properties

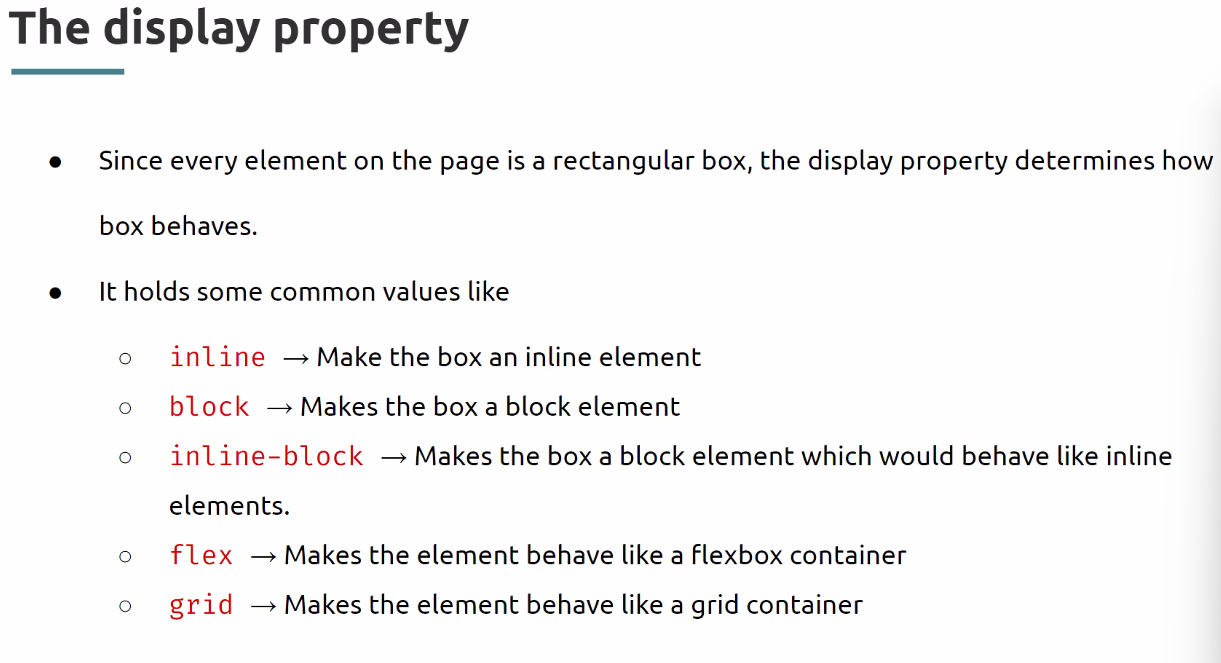
Left means adding space to left side, it will move to right side and vise versa





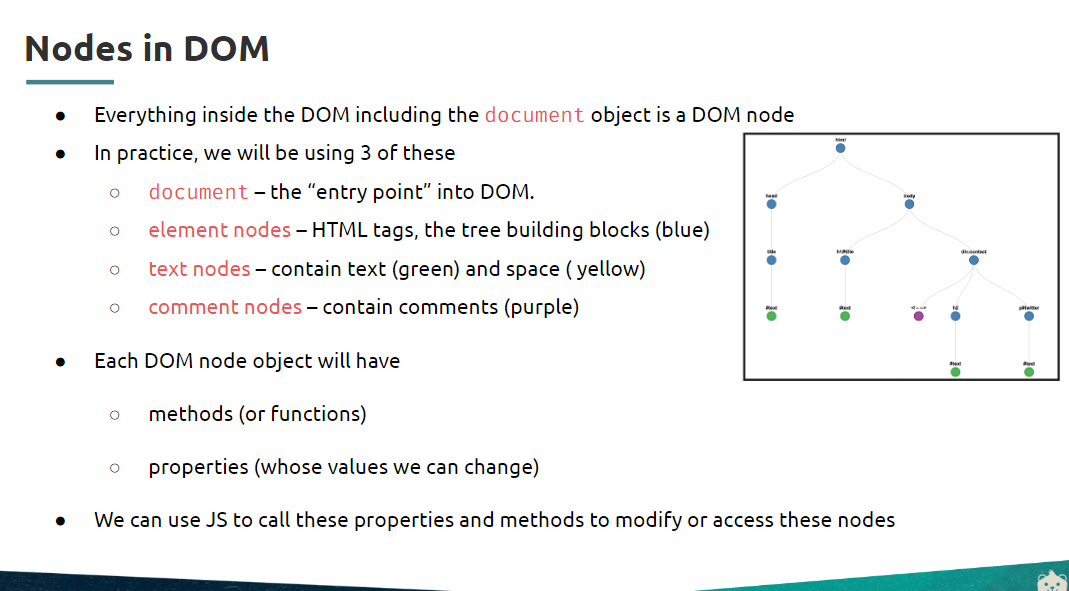
Z-INDEX

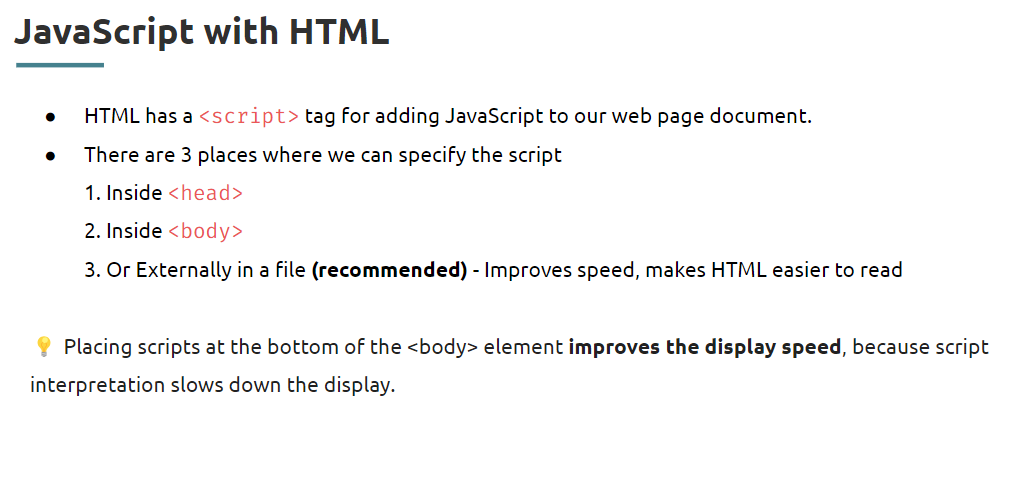
The z-index property in CSS controls the vertical stacking order of elements that overlap. As in, which one appears as if it is physically closer to you. z-index only affects elements that have a [position](https://css-tricks.com/almanac/properties/p/position/) value other than static (the default).



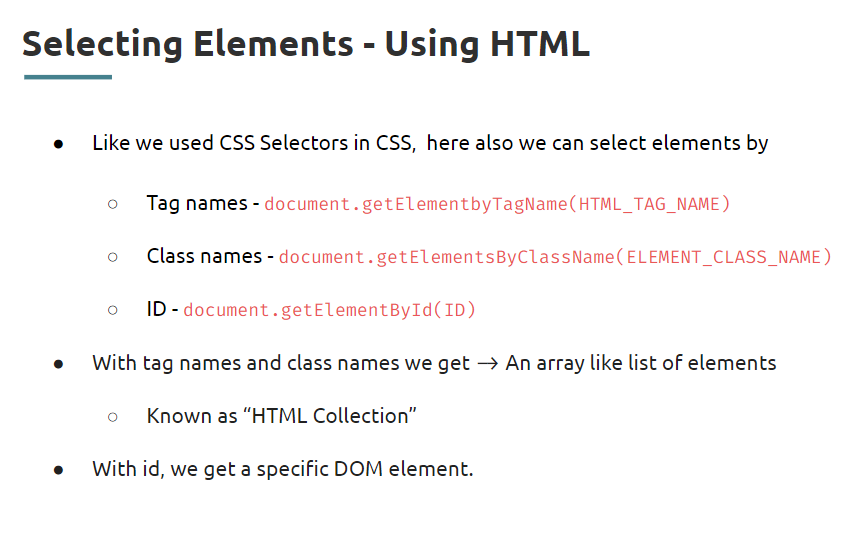
# DOM-JS

DOM is JavaScript representation of html structure and css. Make it possible to manipulate a web page programmatically.





Difference way to Selecting HTML elements for Dom manipulation:



dynamic = document.getElementByID(“id”)

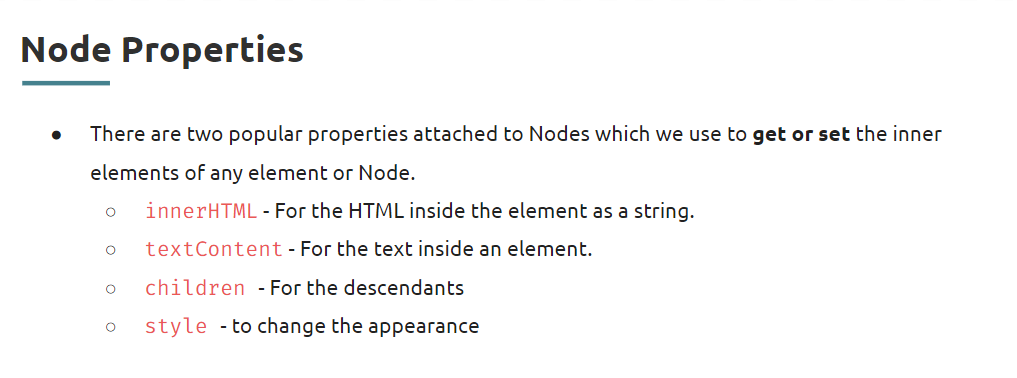
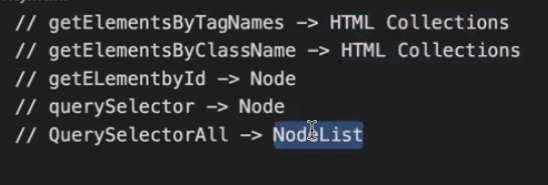
dynamic.innerHTML=”<p>here is the innerHTMl</P>”

dynamic.textContent = “fqsff fwefef”

console.log(dynamic.children)

dynamic.style.backgroundColor=”red”

these styles are act as inline css style, which has highest order of priority

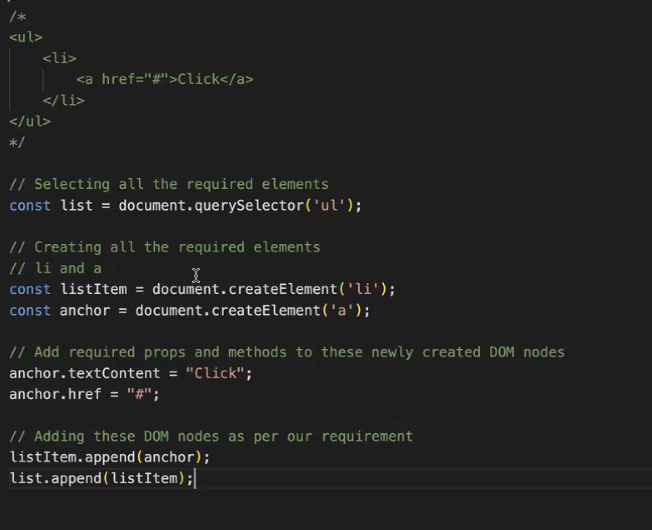


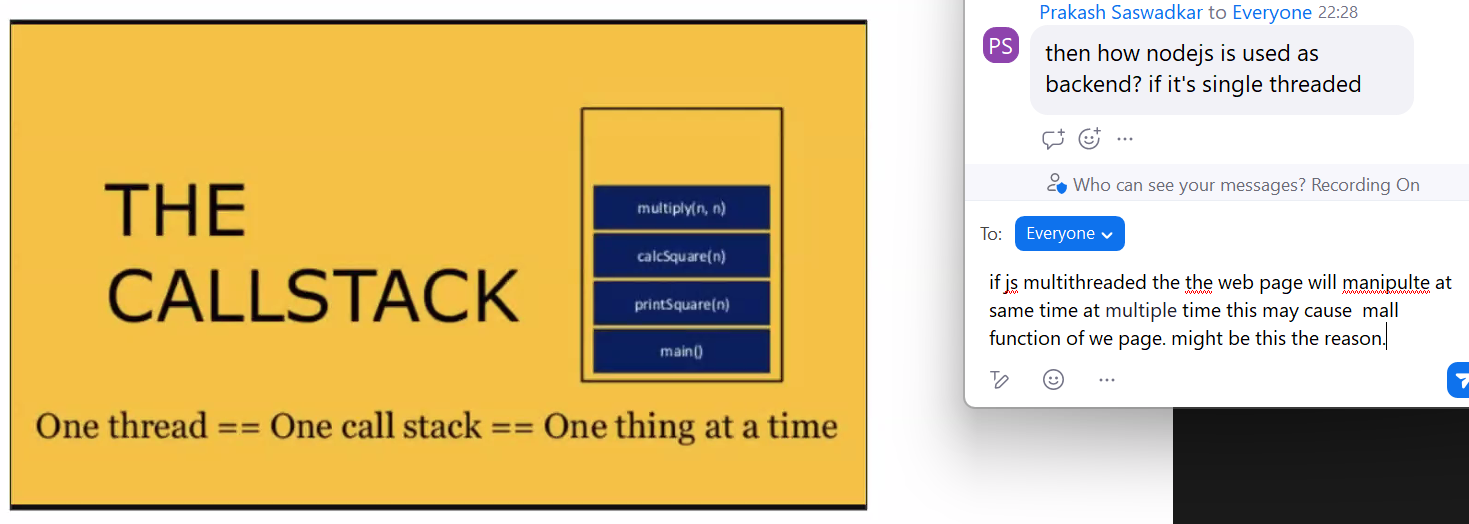
innerHTML and textContent are the setter and getter properties. Childrens is just to retrieve HTMLCOllection

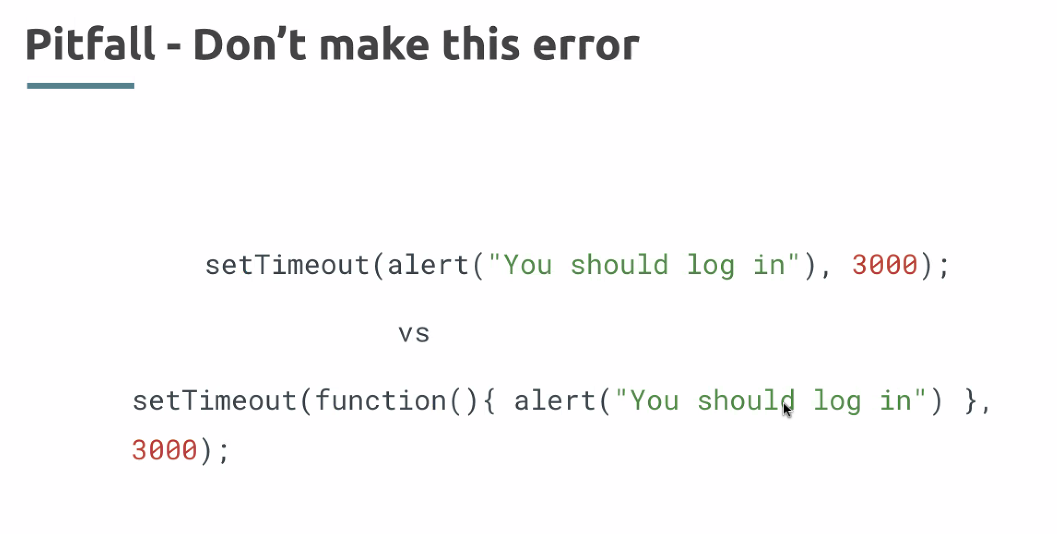
Query selector selectors offers writing css selectors inside it to select the tag of a doc.

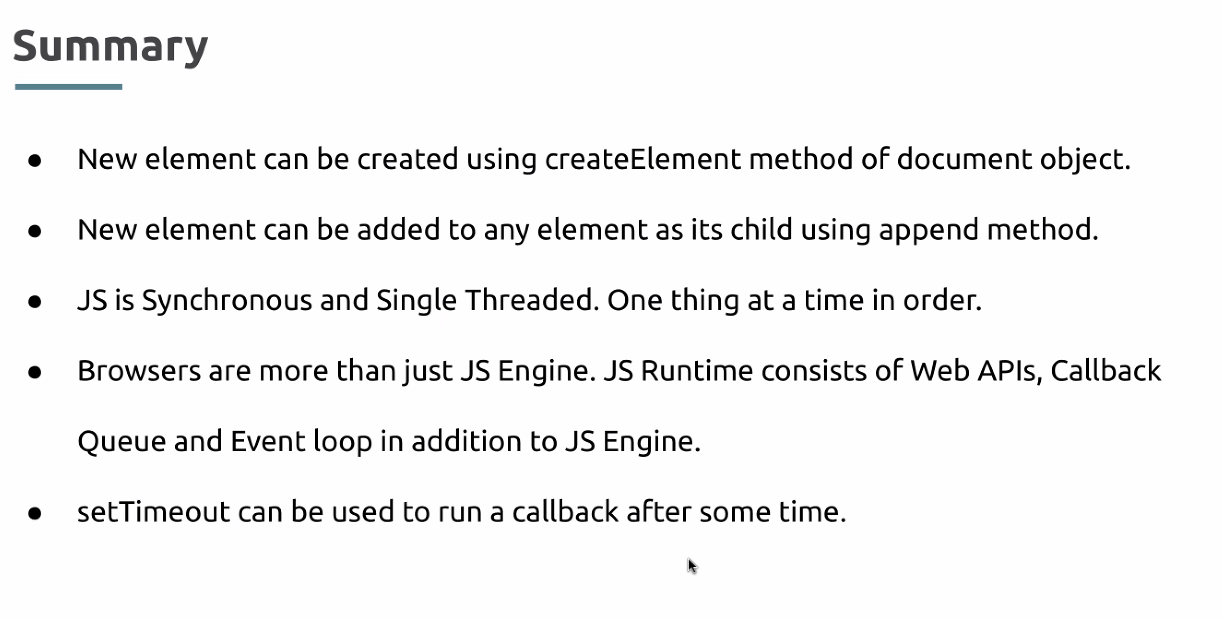
Manipulating dom:

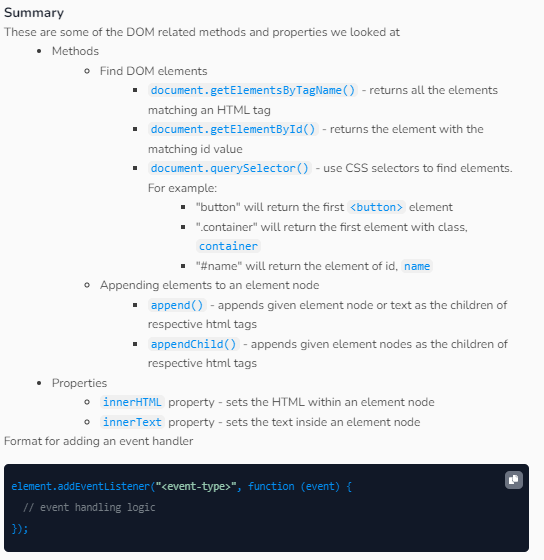
1. Selecting an target element
2. Creating an required element
3. Adding props to it (textContent)
4. Appending it to the target element – [append, appendChild]



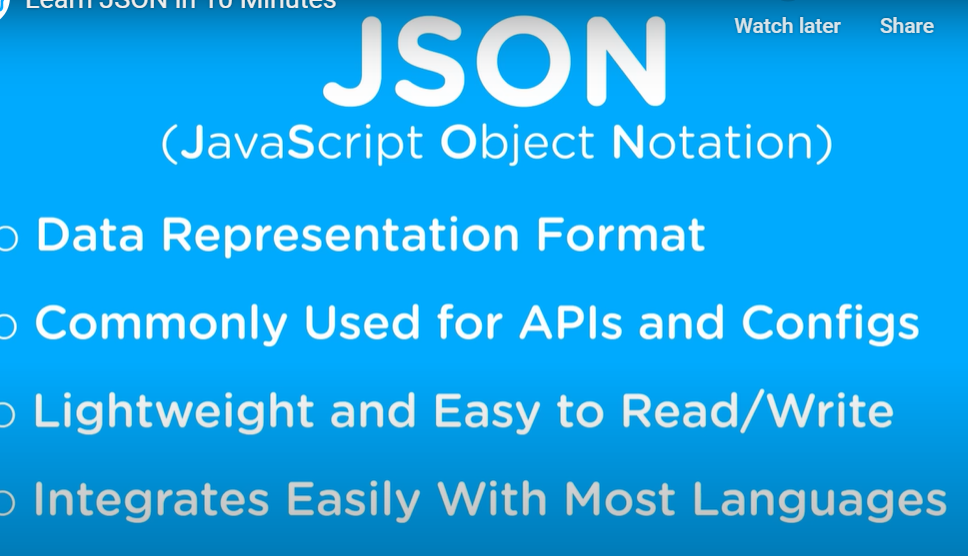




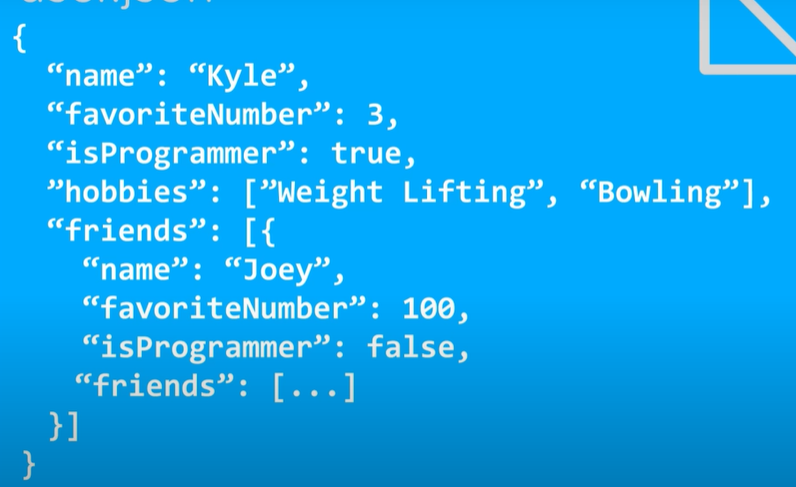




## JSON



JSON type are Strings, Numbers, Boolean, Null, Arrays, and Objects.



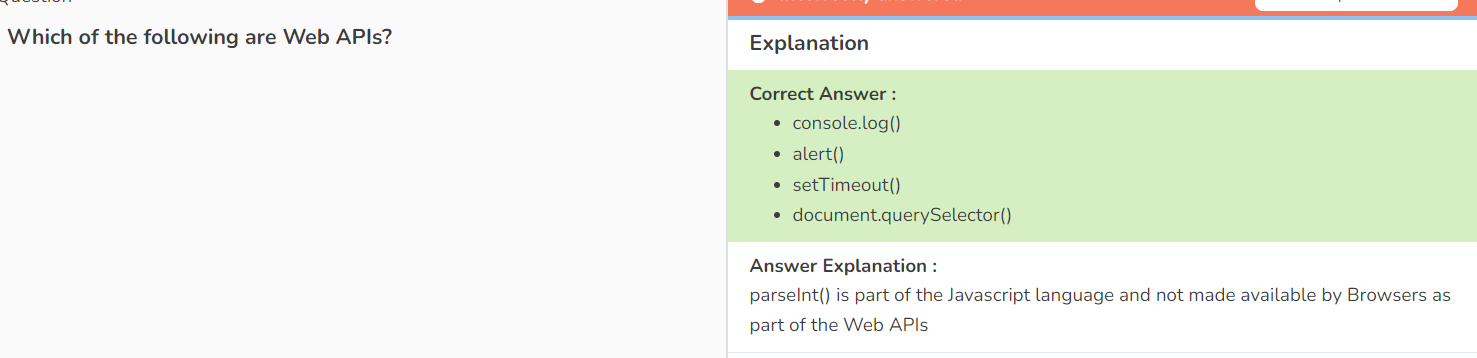
In JS object key can use without double quotes, In JSON key must surround with “”

[JSON.parse()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse) - Parse a piece of string text as JSON, optionally transforming the produced value and its properties, and return the value. You can use the [JSON.parse()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse) method to convert the above JSON string into a JavaScript object

[JSON.stringify()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify) - Return a JSON string corresponding to the specified value, optionally including only certain properties or replacing property values in a user-defined manner. The value to convert to a JSON

### Web API

**I**t Provides the DOM API, FetchAPI, Web Storage API, setTimeOut



#### Fetch API

The [Fetch API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) provides a JavaScript interface for accessing and manipulating parts of the [protocol](https://developer.mozilla.org/en-US/docs/Glossary/Protocol), such as requests and responses. It also provides a global [fetch()](https://developer.mozilla.org/en-US/docs/Web/API/fetch) method that provides an easy, logical way to fetch resources asynchronously across the network.

To overcome the callback hell(nested promises) async and await is used.

#### URLSearchParams API

#### 

<https://developer.chrome.com/blog/urlsearchparams>



params.set('version', 2);

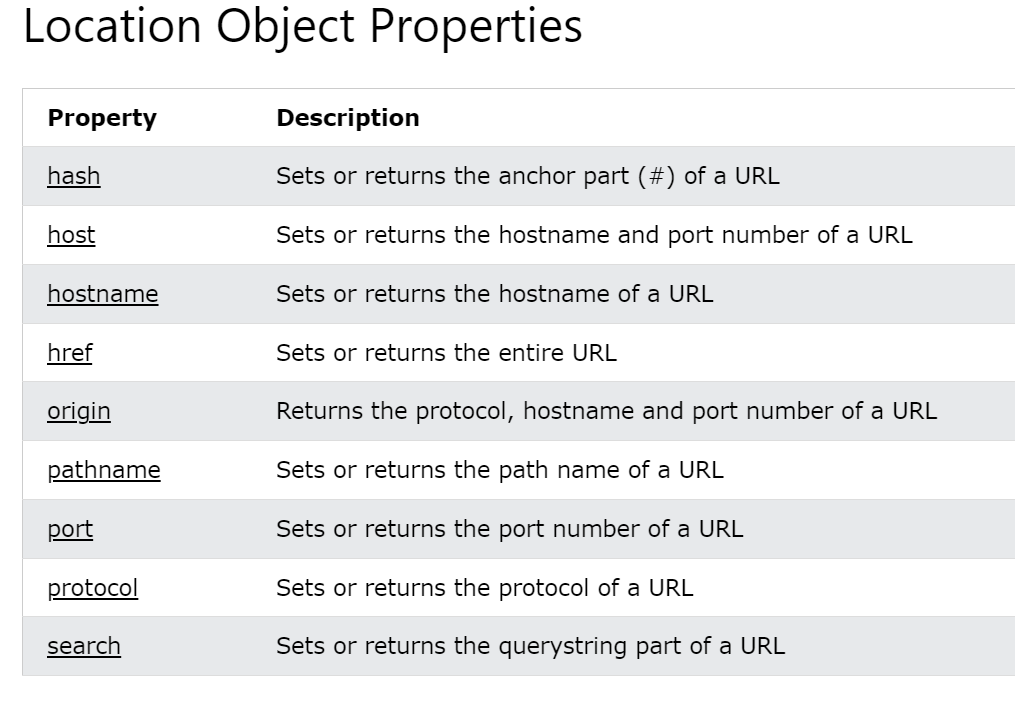
params.append('person', 'Tim');

params.delete('person');

**Window location.search is a alternative way of working with url’s**

The **location object** contains information about the current URL.

let query = location.search; - returns the query parameter including the ?.



# Browser Events



Browser events and events objects:

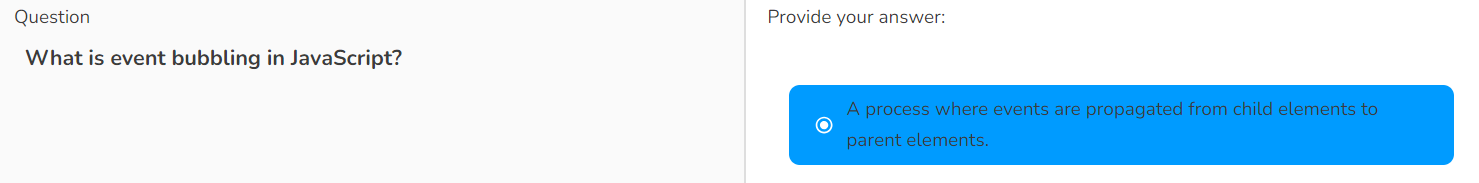
<https://www.freecodecamp.org/news/javascript-events-explained-in-simple-english/>

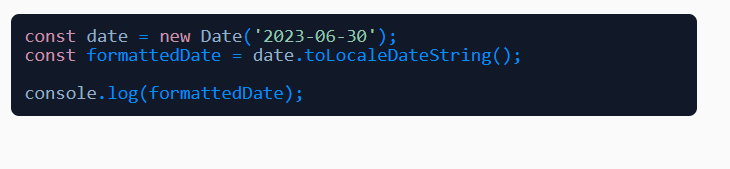
Mouse events, Keyboard events, Touch events, window events, and Form events.

Event Bubbling and Delegation:

All parent elements gets triggers, when child have event.

[](file:///\\EveE)

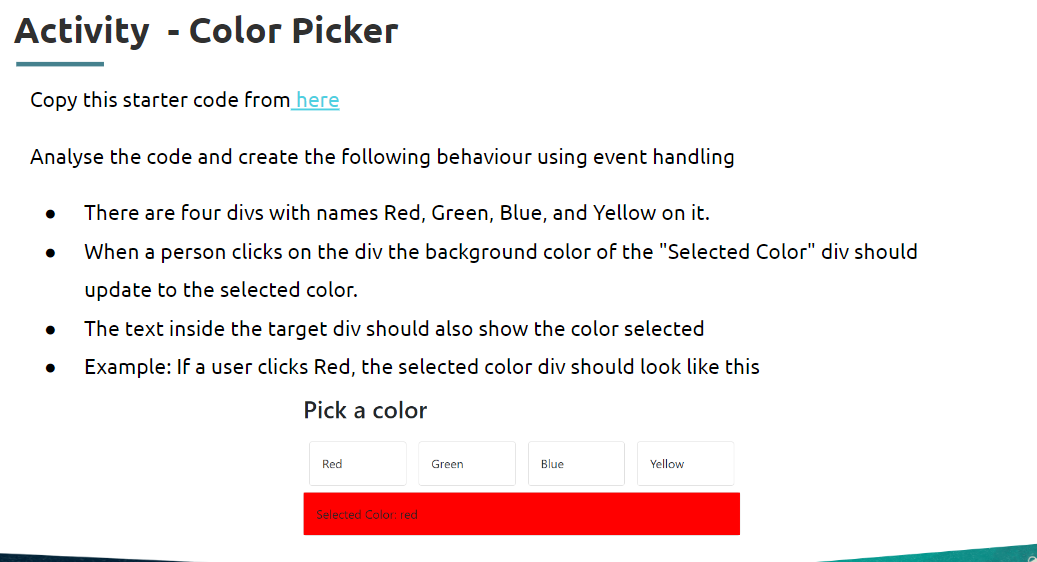




**Event Delegation:**

the whole focus shifts to which element caused the event click, so the target id of child button can access from parent listener.

The below activity one div groups the all color buttons, when ever the a button clicked, evet triggers at div and shares the info of triggered element.



# HTML Forms

form - not only grouping element but also handling the inside elements

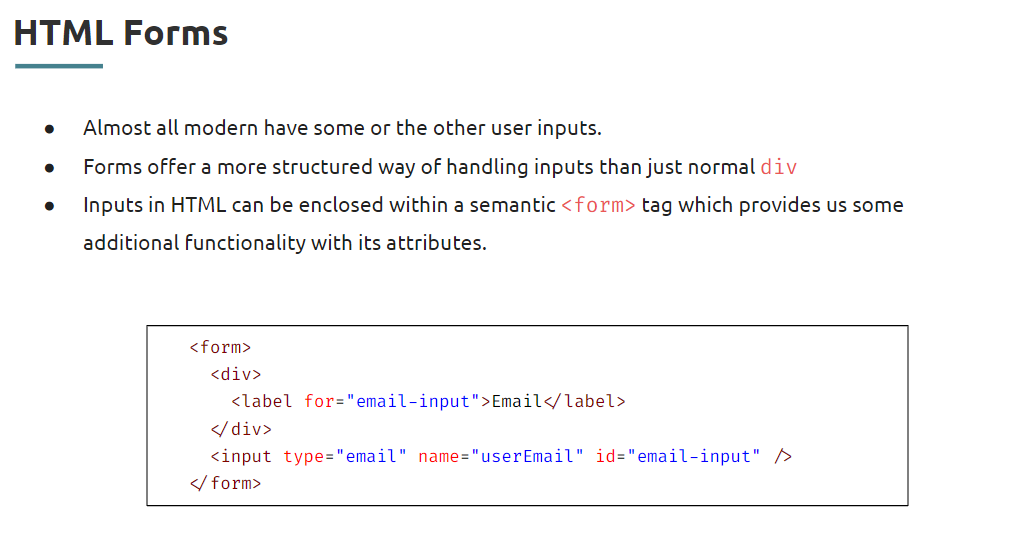
## Working with form submission (POST) , Data retrieve and Validation:

You can store the data in local using local storage.

showing the form data on URL is not a good privacy right? So form action attribute is used to send the data to server.

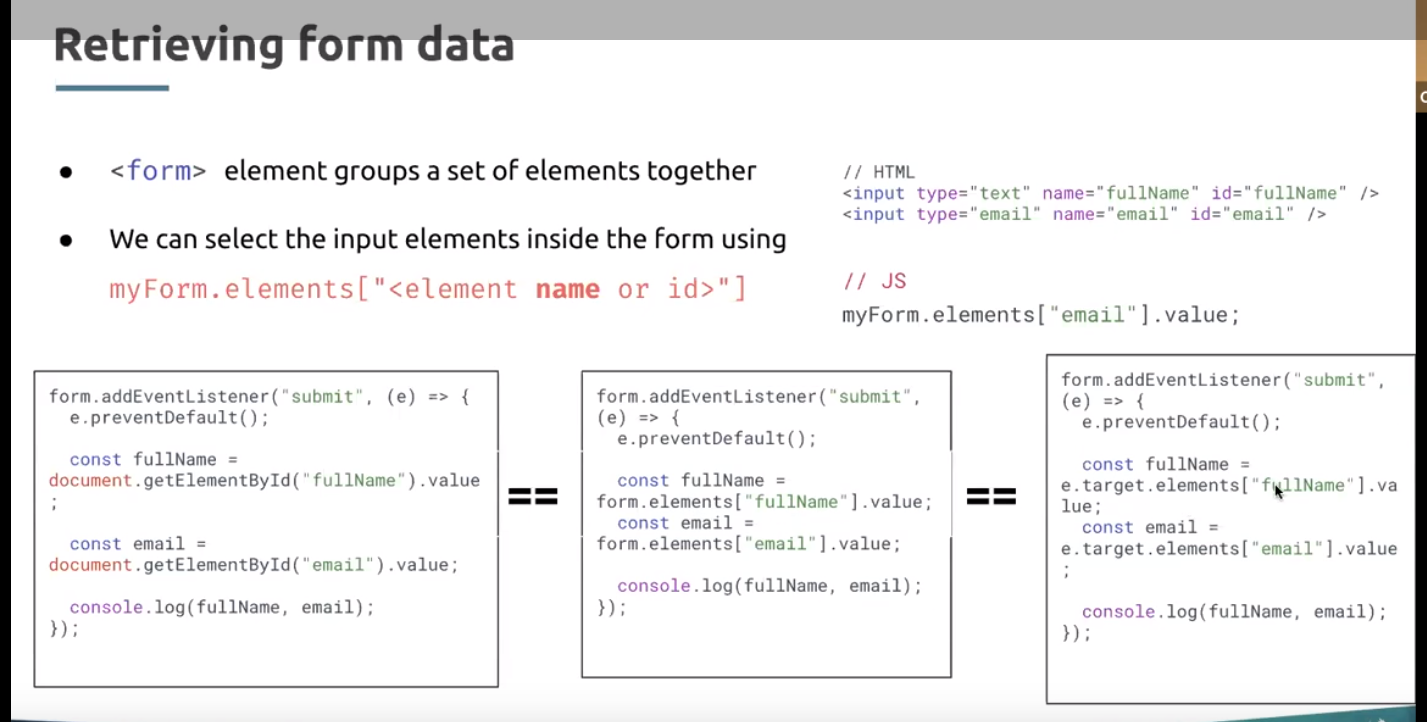
textpassword

radiocheck boxnumberdropdowncalendar



To add relation between label and input tag for attribute is used and the value must be same as id of the input.

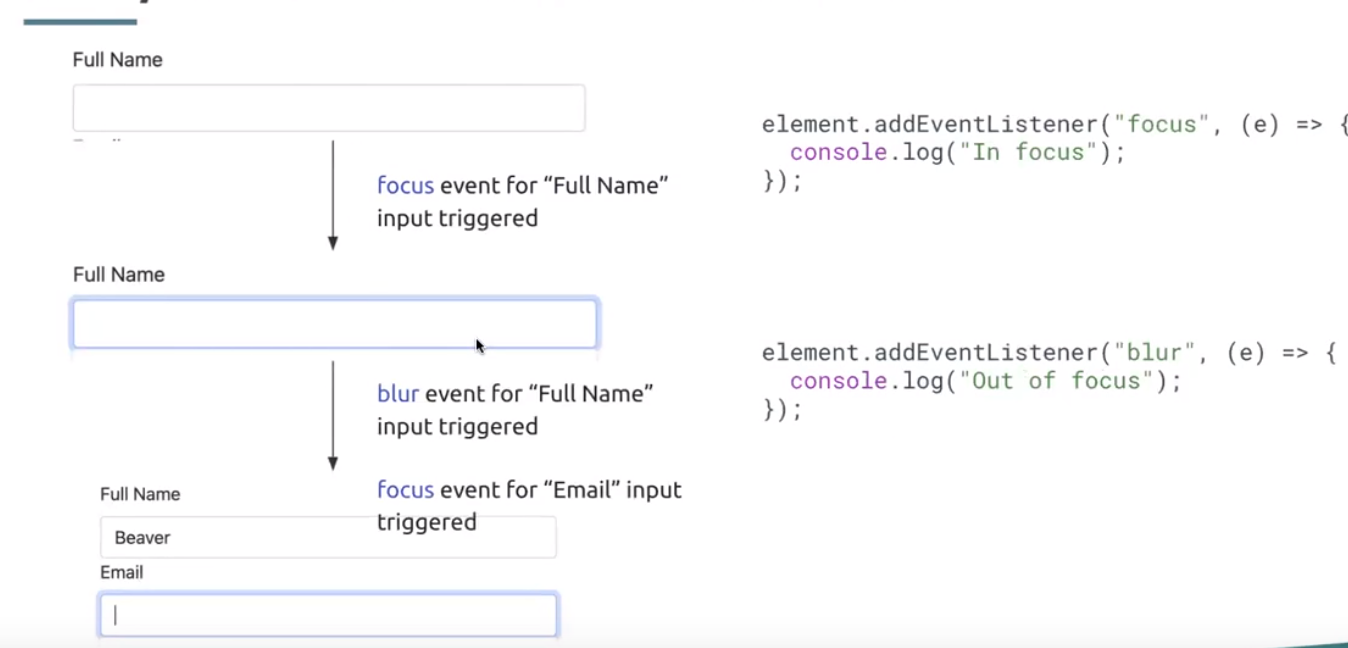
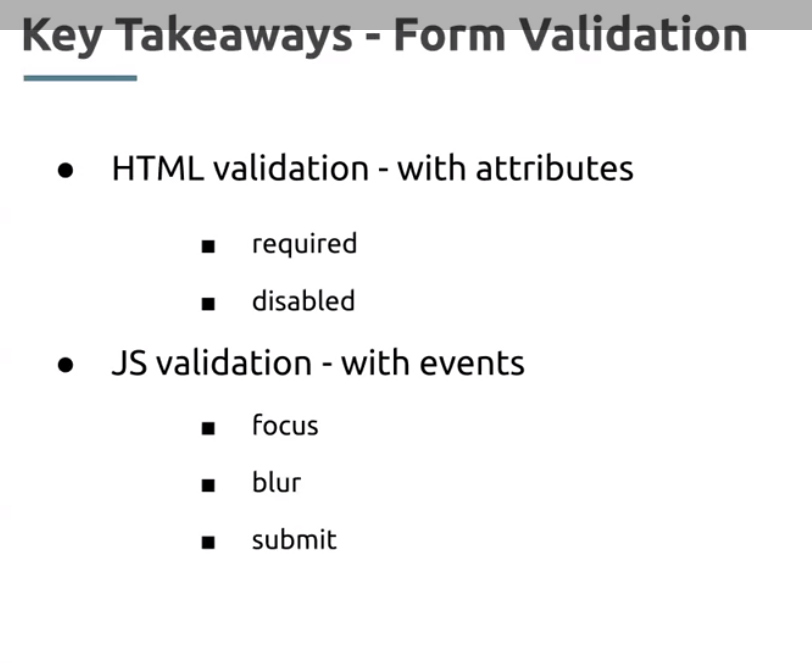
And the name attribute is used to dentify form controls when submitting a form, and to identify same group inputs, to access the value by it. form.elements[name].value



Form Validation:

required, disabled, length- min and max

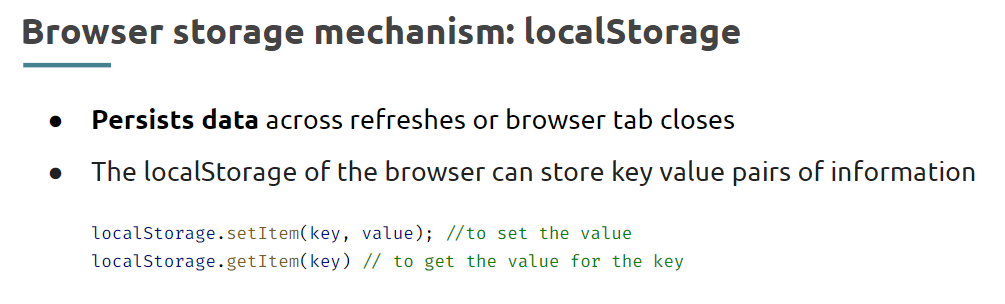
Focus and Blur validation



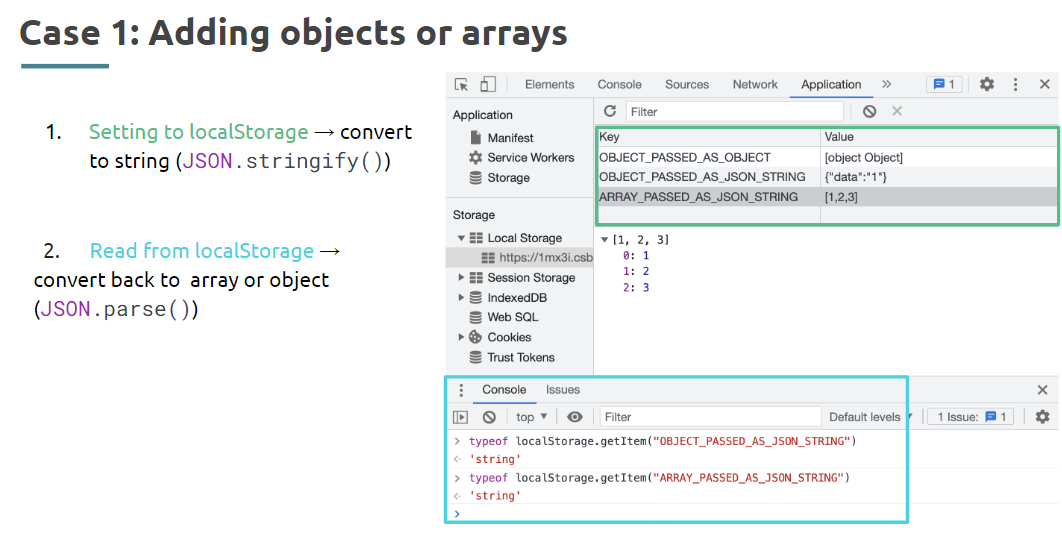
# Local Storage:

Local Storage never deleted from browser until it deleted manually.

localStorage.setItem(key,value) localStorage.getItem() localStorage.deleteItem()



Using the Devtools local storage can view, The Keys and values both are always stored as strings.



The data in local storage are just strings, don’t have toString()to convert into js objects so must use json.stringfy() and json.parse().

<https://help.crio.do/support/solutions/articles/82000902222>

# Web Scraping:



Axios and superagent